ESCUELA POLITÉCNICA SUPERIOR
UNIVERSIDAD CARLOS III DE MADRID

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DESARROLLO DE UNA APLICACIÓN ANDROID DE RECOMENDACIÓN DE PELÍCULAS CON ANÁLISIS DE DATOS EN LA NUBE

Autor: **Iván Benítez Martín**
Tutor: **Jesús Arias Fisteus**

Septiembre de 2018

Agradecimientos

Seguramente este sea el capítulo más difícil de escribir de todo este documento, o al menos el ejercicio de memoria más fuerte donde quieres recordar o mencionar a todas las personas que han pasado por esta época dorada que es la de la universidad.

Lo más sencillo es empezar por la gente que más me ha motivado a la hora de mis estudios y que me han llevado a este momento final.

Los primeros, Iria y Jesús. Sin su dedicación y atención, es muy probable que hubiera tardado un poco más en realizar un proyecto así. Cierro esta etapa de mi vida con un grato recuerdo gracias a personas como ellos, que me han motivado a esforzarme tanto en su asignatura como en este Trabajo de Fin de Grado.

Después, mis padres y mi hermano, que con esfuerzo me han costeado los gastos académicos y que me han apoyado cuando el examen salía mal. Que salían mal.

Aquí se cierra el grupo de gente que me ha agilizado la vida académica y que me han apoyado en los peores momentos. Ahora toca la gente que lo ha hecho increíblemente más difícil.

A Dani, otro genio de la picaresca. Hemos logrado cosas imposibles en ventanas de tiempo increíbles y de una manera fantástica. Nunca nadie ha llegado tan lejos haciendo tan poco.

A Carlos "El Ruso", un genio de la programación, un vago para todo lo demás. Proyectos de desarrollo ha salvado muchos, lo que hacía indispensable su amistad.

Por último, a la Pizzería. Estos fueron los peores. Sin ninguna duda. Pero seguramente haya aprendido más con ellos que con dos masters más. Porque con ellos, la verdadera amistad se mide en la hora a la que te vas a casa.

Resumen

Uno de los principales problemas que están viviendo los servicios de "video bajo demanda" como Netflix, HBO y Movistar+ es la capacidad de mostrar la totalidad del catálogo al usuario. Cada mes, desaparecen y aparecen nuevas películas y series dentro del catálogo, y junto al hecho de que cada vez ese catálogo contiene más y más ítems, el usuario necesita de una ligera ayuda para elegir lo que quiere ver. Ahí entran los sistemas de recomendación que tan avanzados tienen estos servicios. Actualmente se ha demostrado que más del 75% de reproducciones en Netflix han sido potenciadas por el potente motor de sistema de recomendación que tienen, lo que soluciona en parte este problema. ¿Pero, por qué no tener esta ayuda a la hora de ir al cine?

Vivimos en una época de innovación tecnológica, donde ya no solo se está apostando por el desarrollo de dispositivos móviles cada vez más inteligentes, si no que también se está apostando por diferentes vías para facilitar las tareas del usuario así como convertir a estos dispositivos móviles en parte indispensable de la vida del usuario, si no lo es ya. Uno de estos ejemplos es la Inteligencia Artificial.

A lo largo del presente Trabajo de Fin de Grado se ha desarrollado esta idea de explorar las capacidades de la Inteligencia Artificial, profundizando en la tecnología de "*aprendizaje en máquina*" y en los sistemas de recomendación.

Además, todo esto se puede encontrar en una aplicación móvil para Android, acercando al usuario este tipo de experiencias, ya sea por el tamaño de los dispositivos móviles o por su versatilidad.

Abstract

1.- Introduction

The main purposes of this project are the development of a recommender system on the cloud and the mobile application that uses this service, which it will be adapted in order to look as a user library where the user will be able to save their favorite movies and will be able to discover the new movies the system will recommend him.

Twelve years ago, Netflix was willing to experiment with new and innovative approaches and opened a contest in order to award the new algorithm that would improve its recommender system, *Cinematch*. This kind of contest was pioneer in the search for innovation in this type of sector.

In this project, one of the main goals is to achieve positive results regarding to the recommender system to improve the ones achieved in the Netflix contest 12 years ago. Therefore, the “on the cloud” technology is presented as an ideal platform to build, develop and deploy the machine learning technology, and the cloud platform chosen has been Microsoft Azure.

Regarding to the mobile application, the idea was to develop, in an Android environment, a system similar to a social network, with the goal of

presenting a nice and intuitive design. Through an authentication system, the system will give to the user the access to the database of TMDb and will allow the user to explore new movies individually, lists of favorite movies of the users, lists of most rated movies and the personalized recommendations for the user. In order to integrate this recommender system, the cloud platform will be raised as API in which by means of calls to the web service will be possible to obtain the identifiers of the recommended films.

2.- Work Plan and Budget

The different phases in which the work plan has been developed are listed below:

- Phase 1: Study of the problem and documentation.
- Phase 2: Collection of requirements.
- Phase 3: Establishment of the technologies that will be used.
- Phase 4: Design of the recommender system and the mobile application.
- Phase 5: Integration and implementation of both systems.
- Phase 6: Test plan.
- Phase 7: Documentation.

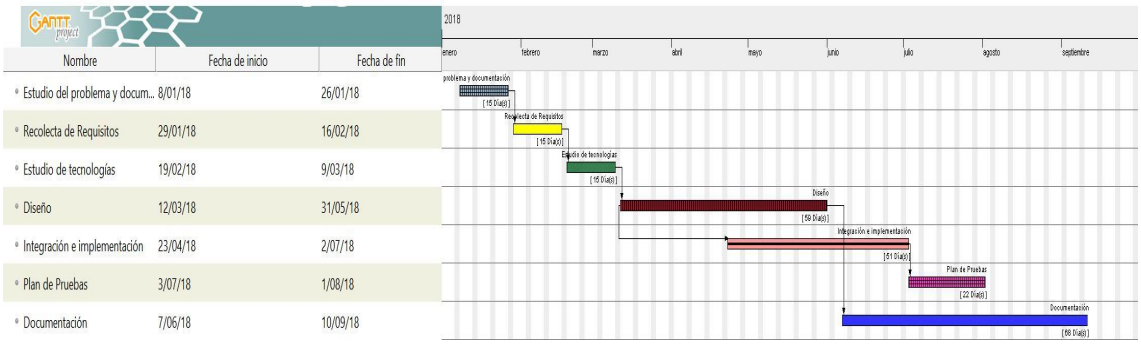

Figure 2.1: Gantt's Diagram

In order to present the final budget, some topics must be discussed before. The fees will correspond to a bachelor Graduated of Engineering in Telecommunications Engineering. The development of the project has taken eight months, with 160 days and a total amount of hours equivalent to 400 hours. As the project supervisor, the fees will correspond to a Telecommunications Engineer whose total amount of hours are 30 hours, between all planned meetings and on-line supervision of work.

The final cost breakdown is presented below:

Reason of the cost	Cost
Cost of labor personnel	9611,55 €
Material Cost	233,33 €
Software with amortization	88,88 €
Cloud Service cost	60,76 €
Indirect costs (20% of all above)	1998,90 €
IVA (30% of all above)	3598,03 €
TOTAL	15591,46 €

Table 2.5: Final Cost Breakdown

3.- Requirements

In this chapter, the software requirements considered for this project will be presented. Before that, just to clarify, there's been a distinction between three types of requirements:

- Functional requirements: the system would be unstable if they are not achieved.
- Restriction requirements: they form the behavior the system must obey.
- Non-functional requirements: correspond to the definition of quality attributes related to system properties.

The software requirements are:

Identifier	Name	Description	Type of requirement
RS-01	API REST TMDb	The Android Application will provide a communication with the TMDb database by a web service communication.	Functional requirements
RS-02	REST API Recommender System	The application will need a web service connected to the recommender system located on Azure.	Functional requirements
RS-03	Mobile Internet Connection	The application will not be able to start if there's no mobile internet connection.	Functional requirements
RS-04	Permissions for third-party applications	Permissions of the TMDb database for the use of the	Functional requirements

		user data. If not, the user won't be able to access to the full list of features the application provides.	
RS-05	Recommender System	The application will provide to the user a recommender system.	Restriction requirements
RS-06	Authentication System	The user will be able to identify himself, even as a user or as a guest.	Restriction requirements
RS-07	Favorite movie galleries, best rated movies or movie lists	The user will obtain a movie gallery by choosing any of the main menu features.	Restriction requirements
RS-08	Access to the movie's crew list	The user will be able to find the cast and crew of the movie.	Restriction requirements
RS-09	Storage system	The application use SQLite files that require an important amount of internal memory needed to run some features.	Restriction requirements
RS-10	Search Feature	The user will be able to search a movie in the TMDb database.	Restriction requirements
RS-11	Save favorite movies in SQLite files	The system will be able to recognize if a movie is already in the user's favorite list, indicating it on screen	Non-functional requirements
RS-12	Correlation Algorithm	In order to give a movie recommendation to the user, the application will look in a stored	Non-functional requirements

		SQLite file if there's a MovieLens identififer that's related to the TMDb identifier.	
RS-13	Android development	Android Studio will be used as the main development environment and platform.	Non-functional requirements

Table E3.01: Software Requirements

4.- System Architecture

In this chapter, the reader will be able to find a high layer description of the developed systems and their communications between them.

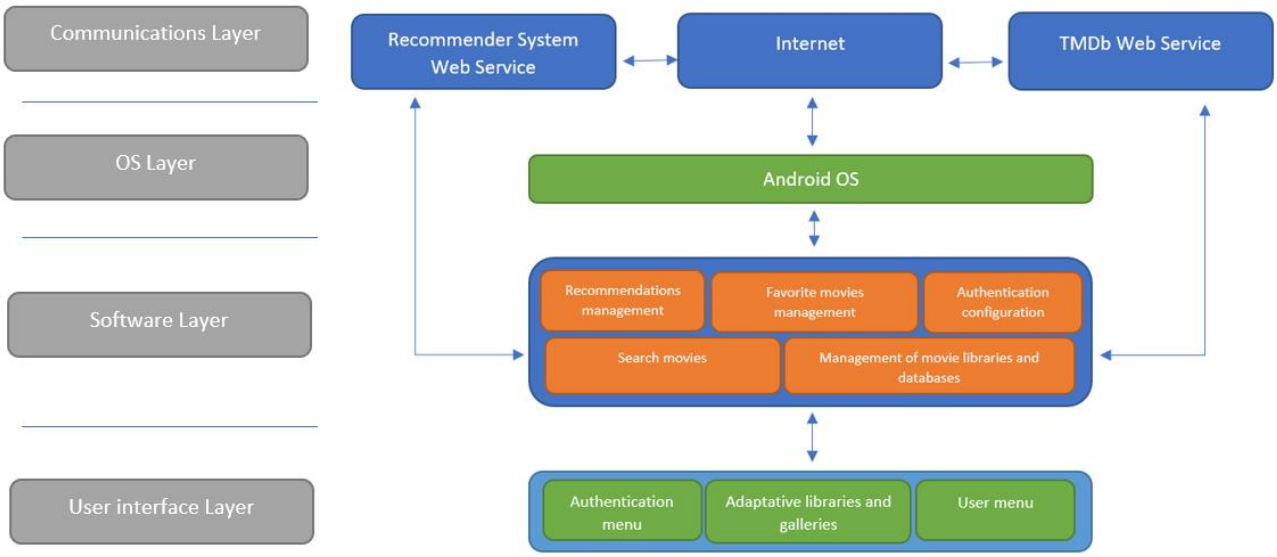

Figure E4.01: System Architecture

5.- Design

This chapter will be useful to understand the design of the three main elements of the project's architecture: the Azure Recommender system, the Android mobile application and the TMDb web service.

The recommender system is composed of two different schemes. First of the is the training model, where the system is designed to train the MovieLens datasets and is presented below:

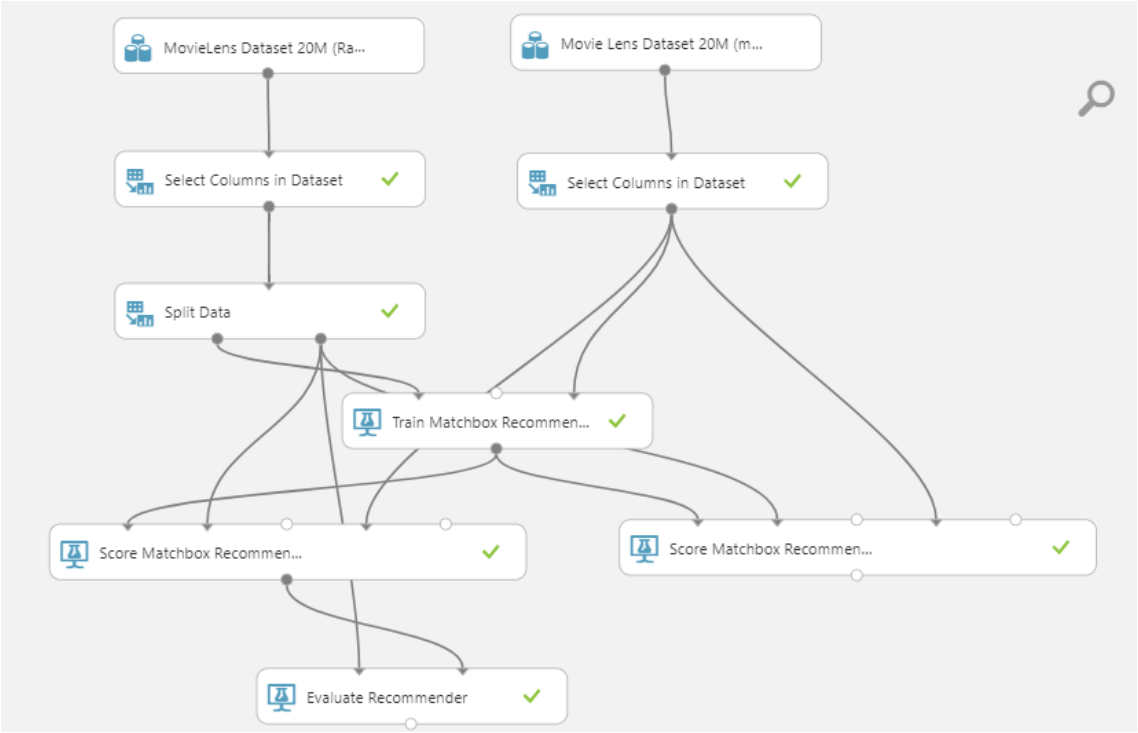

Figure 6.1: Training model

As it is presented on the top of the figure, the two datasets will be processed to chose the desired columns of this sets, split its data in two segments (one to be trained and the other one to evaluate our model), the Train Matchbox Recommender module, which reads the related datasets with user-item-rating and gives back a recommender system module, and the last module, the Evaluate Recommender module, that will allow the system to evaluate the rating prediction of the items as the precision of the recommendations. In this case, the system has been evaluated with the next metrics:

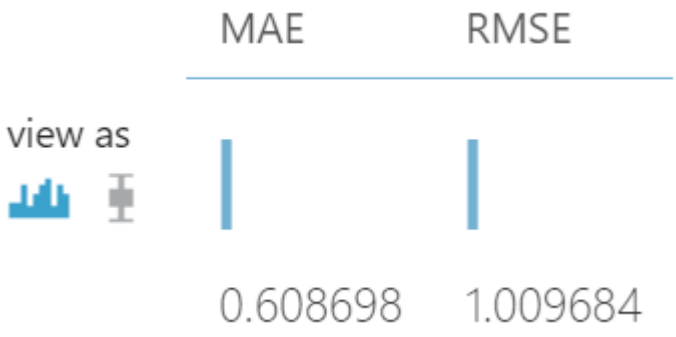

Figure 6.8: Evaluation metrics

As the application architecture, different modules have composed its architecture, as they are presented below:

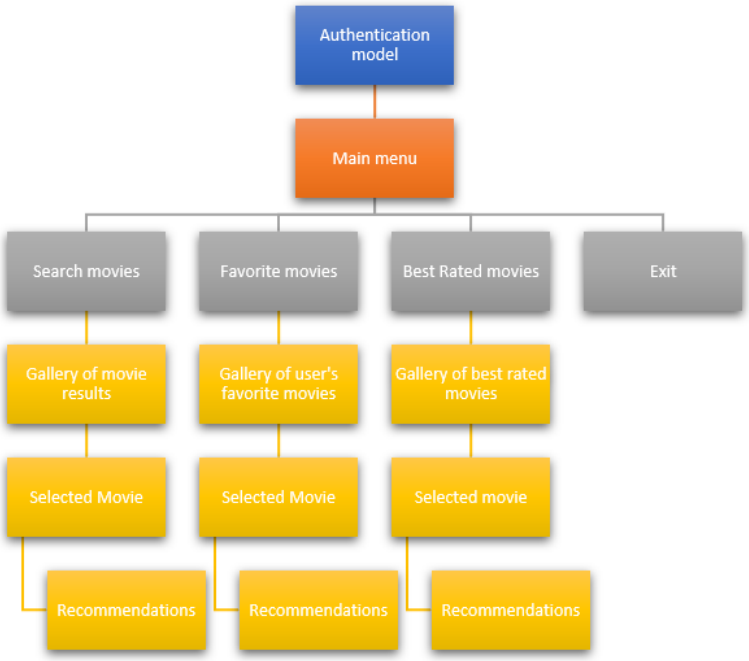

Figure 6.21: Diagram of the application's architecture

6.- Implementation

In this chapter, the recommender system web service designed in Azure is presented with the next architecture:

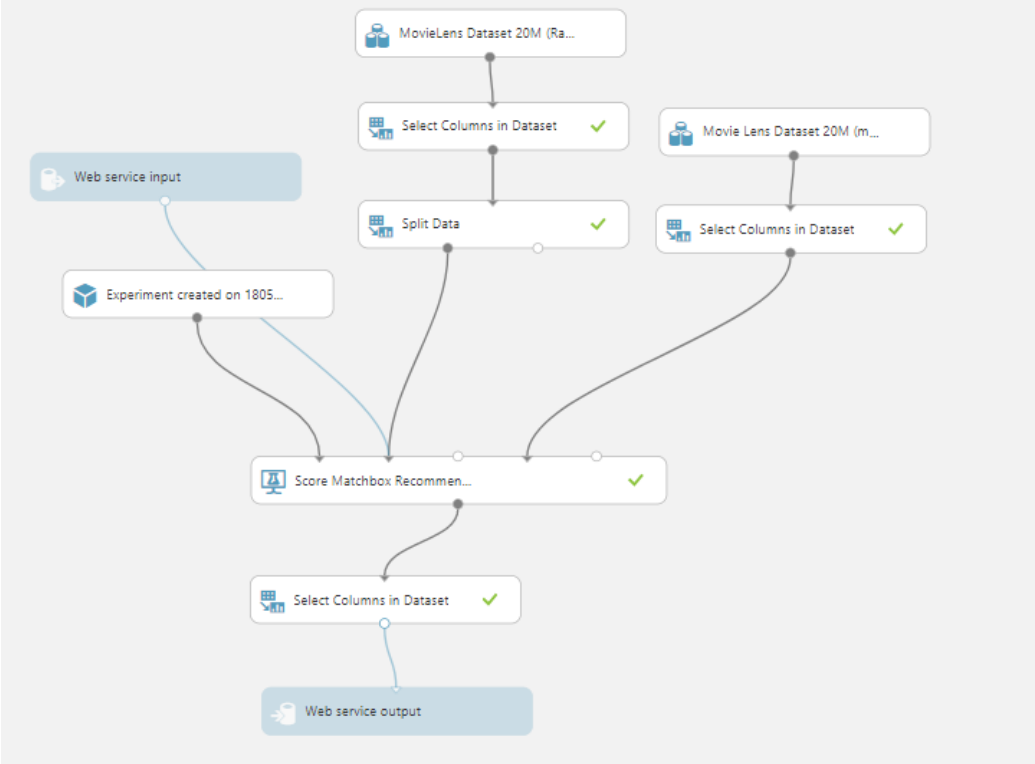

Figure 7.1: Architecture of the Recommender system's web service

Regarding to the TMDb web service, the “Retrofit” library has been used in order to list the different calls to the database as separate Java interfaces. Therefore, “Retrofit” is a REST client that will allow to make requests of any kind, getting a response as a Plain Old Java Object (POJO).

7.- Test Plan

A brief scheme of the different test cases that have been performed in the system is presented below:

Test Case	Requirements tested	Description
TC-01	RS-02 and RS-05	The input will be a movie and Azure will give back a list of six recommended movies according to the input.
TC-02	RS-06, RS-03 and RS-01	The user will complete the authentication process: to receive a request token, to accept the permissions for third-party applications and to introduce the username and password, receiving a session identification to continue.
TC-03	RS-11, RS-07 and RS-01	The user will press in the main menu the button of “My favourite films” and will obtain the gallery of the films and the results will be registered in a SQLite file.
TC-04	RS-08. RS-01 and RS-11	The user will press one of the items of the gallery. A new activity will show the crew and cast detailed page of the film.
TC-05	RS-01, RS-02, RS-05, RS-07, RS-09 and RS-12	The user presses the button of “Ivan’s Recommendations”, which leads the user to a gallery of recommended movies according to

		TMDb or the developed Recommender System.
TC-06	RS-01, RS-07 y RS-10	The user will press the button of "Search Movie", will introduce the name of the item and will press the button to launch the searching and obtain the desired movie results.

Table E7.01: Test Cases

8.- Conclusions

From the original software requirements, a complete resolution has been achieved. Machine Learning technology has been developed during decades with excellent results regarding to the dataset management, analytics, build and refine a model and make it accessible to the user. Even Azure allows a fast development considering this kind of projects, one improvement listed is to build and deploy a native recommender system, which would change the different communications between the systems, but would allow to explore in a deeper way the Machine Learning technologies, adding more personality to the project.

The use of TMDb web services has accelerated the project in a considerable way, allowing this association a series of very interesting features, such as the administration of roles, authentication or item management between both portals.

One of the milestones of the projects has been the creation of different processes that allow to combine REST requests with the asynchronous calls that Android allows.

Finally, another milestone has been the training in mobile applications development, which motivates to be trained in the development in other types of platform as iOS, which is considered as well as another improvement to elaborate in the future.

Contenidos

Índice de figuras

Índice de ecuaciones y tablas

Capítulo 1: Introducción

1.1.- Motivación

En enero de 2016, el término “inteligencia artificial” no era uno de los 100 términos más buscados en Internet. Sin embargo, no significa que no se hayan conseguido avances en este estudio. El aprendizaje en máquina lleva siendo una rama puntera de la inteligencia artificial la última década. ¿Qué es lo que lleva a que, en mayo de 2017, este término antes mencionado se posicionara en el número 7 de términos más buscados en motores de búsqueda? ¿En qué es posible basar el incremento de popularidad que está refiriendo?

Este pico de popularidad mencionado está actualmente empujando a multitud de vendedores de software a introducir la IA en su estrategia de producto. Multitud de clientes ven absolutamente necesario incluir en su estrategia de negocio digital esta tecnología. Pero lo que provoca esto es más confusión, ya que se está incrementando esta oferta de productos con IA sin ninguna diferenciación real. A fecha de escritura de este proyecto, no es difícil observar en multitud de anuncios comerciales como la última televisión de X marca ha introducido de forma novedosa la inteligencia artificial. O cómo el último modelo de teléfono de la empresa tecnológica Y ha conseguido integrar de forma exitosa esta tecnología.

Es necesario olvidar ese tipo de titulares sensacionalistas con ideas como "los robots están robando nuestros trabajos". El desarrollo, la innovación y el uso de herramientas para facilitar la vida son pilares que sostienen el progreso de la humanidad a lo largo de la historia. Tecnológicamente, esta época que nos ocupa está viniendo a ser una revolución de datos en la cual, a medida que se logren esos progresos en esta rama, roles y trabajos cambiarán, creando posiciones nuevas, nuevos modelos de negocio, etc. En el software en nuestros teléfonos, en el trabajo, como herramienta de acceso a la información e incluso en nuestros hogares y automóviles, la inteligencia artificial ha llegado para quedarse. De acuerdo con Gartner [1], en 2020 "casi todos los nuevos productos de software" tendrán integradas las tecnologías de aprendizaje automático o "Machine Learning".

En el campo del entretenimiento, hace 12 años, Netflix, deseosa de experimentar con nuevos enfoques, abrió un concurso para premiar al algoritmo que mejorara el sistema de recomendación que poseía en su día, *Cinematch*. ¿El premio? 1 millón de dólares. Llega a resultar muy interesante como este concurso fue pionero en la búsqueda de la innovación en este tipo de sectores, abriéndole la puerta a otras plataformas para desarrollo como Kaggle o TopCoders. Como se comentará a continuación más detalladamente en el apartado "Estudio del Arte", *FilmAffinity* desarrolló, en su origen como portal, un sistema de recomendación basado en la afinidad de los usuarios de ese portal. Así, este portal se convirtió en un lugar clave para todo aficionado al cine.

Por tanto, estos dos ejemplos mencionados son dos de las grandes inspiraciones que motivan al autor de este proyecto a desarrollar lo que se muestra a continuación. Durante las fases iniciales de este proyecto, la investigación en aprendizaje en máquina, el desarrollo de aplicaciones móviles y la aspiración a lograr una implementación relacionada con el mundo del cine se convirtieron en los pilares para, en primer lugar, la construcción de un sistema de recomendación que buscara la mayor afinidad entre usuario y película, y en segundo lugar, para lograr una mayor interacción por parte del usuario mediante el uso de una aplicación Android.

1.2.- Objetivos

Los propósitos principales de este proyecto son el desarrollo de un sistema de recomendación en la nube y de una aplicación móvil que utilice dicho servicio. Esta última servirá de biblioteca particular en la que cada usuario podrá almacenar sus películas favoritas y descubrir nuevas películas que el sistema recomiende.

En cuanto al sistema de recomendación, se intentarán lograr unos resultados positivos que mejoren los conseguidos por aquel sistema de recomendación ganador del concurso de Netflix que se ha mencionado con anterioridad. Así, la tecnología en la nube se presenta como una plataforma ideal

[1] Gartner Says AI Technologies Will Be in Almost Every New Software Product by 2020. (2018). Retrieved from https://www.gartner.com/newsroom/id/3763265

en la que construir y desarrollar la tecnología de aprendizaje en máquina. Esa plataforma en la nube elegida ha sido Microsoft Azure. A su vez, se ha conseguido una base de datos amplia y completa que se utilizará como entrenamiento para nuestro sistema. Se han diferenciado dos tablas distintas en la que se recogen las valoraciones de miles de usuarios, formando una comunidad, sobre 1 millón de películas registradas junto a su género, su año de estreno y su identificador en otras bases de datos más conocidas como son IMDb[2] o TMDb[3]. Así, el usuario podrá marcar, dentro de la aplicación, una película que le haya gustado y este sistema le devolverá hasta seis títulos diferenciados que pueden que sea de su agrado.

Respecto a la aplicación móvil, se ha desarrollado en un entorno Android un sistema similar a una red social cuya interfaz gráfica intentará ser lo más agradable e intuitiva posible para el usuario. En este sistema se permitirá acceso, mediante usuario y contraseña, a la base de datos que dispone TMDb y, en ella, será posible acceder a películas individualmente, listas de películas favoritas del usuario, listas de películas más valoradas por la comunidad de usuarios de TMDb y las propias recomendaciones de cada película. Para la integración del sistema de recomendación, la nube se planteará como API en la que mediante llamadas al servicio web será posible conseguir los identificadores de las películas recomendadas. Mediante el recurso que proporciona el servicio web de TMDb, estos identificadores otorgarán todas las características necesarias para mostrar la información de la película al usuario.

En resumen, algunos de los objetivos que se desean cumplir a lo largo de este trabajo son:

- Gestionar mediante un servicio web y una interfaz gráfica adecuada, un cliente móvil a la base de datos TMDb.
- Desarrollar un sistema de recomendación en la nube e implantarlo como servicio web.
- Una aplicación Android accesible y atractiva al usuario en el que pueda organizar su filmoteca particular.

1.3.- Estructura de la memoria

La organización en capítulos se dispone de la siguiente manera:

- Capítulo 2: Plan de trabajo y presupuestos. A lo largo de este capítulo se prepararán las fases del proyecto que se han trabajado a lo largo de la duración del mismo. Además se presentarán un presupuesto para el

[2] https://www.imdb.com/
[3] https://www.themoviedb.org/

proyecto (el coste del material, el software con amortización, el personal, entre otros)

- Capítulo 3: Estado del arte. En este capítulo se exponen todas las tecnologías que se han tratado durante el transcurso del proyecto y se detallan sus características.
- Capítulo 4: Requisitos. Aquí se muestran los requisitos iniciales de software que se han querido implementar como objetivos iniciales del proyecto.
- Capítulo 5: Arquitectura del sistema. En este capítulo se muestra la comunicación entre los bloques del sistema, mostrando un vistazo general al sistema en sus más alto nivel.
- Capítulo 6: Diseño. Al más bajo nivel se irán mostrando los módulos que componen el sistema del proyecto, así como sus clases principales, las estructuras de los conjuntos de datos, las soluciones de terceros a integrar o la interacción con el usuario.
- Capítulo 7: Implementación. A lo largo de este capítulo se detallan los aspectos más relevantes a la hora de implementar las soluciones al proyecto.
- Capítulo 8: Plan de pruebas. En este capítulo se han detallado las distintas validaciones realizadas al sistema para comprobar que los requisitos iniciales se cumplen a la perfección.
- Capítulo 9: Conclusiones. De presentan una serie de valoraciones finales que han surgido a lo largo del proyecto y se enumeran algunas de las posibles ideas para trabajos futuros sobre este proyecto.

Capítulo 2: Plan de trabajo y presupuesto

2.1- Plan de trabajo

Las fases en las que se ha desarrollado el plan de trabajo se presentan a continuación:

- Fase 1: Estudio del problema y documentación pertinente. Durante esta fase, los objetivos a cumplir ha sido una profunda investigación sobre el aprendizaje en máquina y desarrollo de aplicaciones Android. Este estudio parte desde sistemas de recomendación hasta los lenguajes más comunes en los que se desarrollan estas tecnologías. Además, se planteará la idea de construir el sistema de '*Machine Learning*' o 'aprendizaje en máquina' mediante la computación en la nube y un estudio de qué tecnologías de '*cloud computing*' o 'servicio en la nube' hay disponibles en mercado.
- Fase 2: Recolecta de requisitos. Esta fase comprende la identificación de las características funcionales que, tanto la aplicación como el sistema de recomendación, deben de cumplir. Es necesario el análisis de los recursos de tiempo y de las necesidades del proyecto que se disponen para completar un proyecto de estas medidas.

- Fase 3: Establecimiento de las diferentes tecnologías a trabajar. Después de la segunda fase y tras haber identificado las mejores opciones para la construcción del sistema de recomendación, se elegirá la más adecuada. Para ello, serán necesarias diversas pruebas en las que se recojan los diferentes resultados que se obtendrían. A su vez, después de una correcta evaluación de las tecnologías actuales que lleven a cabo la computación en la nube, se establecerá la opción más accesible y adecuada.
- Fase 4: Diseño del sistema de recomendación y de la aplicación móvil. Esta fase comprende la arquitectura del sistema de recomendación y de la aplicación móvil, así como sus diseños. Su importancia es alta, puesto que las decisiones a tomar serán claves para el posterior desarrollo del proyecto.
- Fase 5: Integración e implementación de ambos sistemas. Tras el diseño individualizado de cada sistema, la implementación será el siguiente paso bajo la observación de los requisitos iniciales recogidos en anteriores fases. A continuación, el siguiente paso sería la integración de ambos sistemas.
- Fase 6: Plan de pruebas. El penúltimo paso abarcará la evaluación del sistema al completo: su funcionamiento, características, etc. Los resultados finales servirán de estudio para comprobar el alcance a la hora de cumplir los requisitos especificados inicialmente.
- Fase 7: Documentación. Como punto final, esta fase abarcará la escritura del documento que engloba la descripción del proyecto y de los resultados obtenidos durante el desarrollo de este.

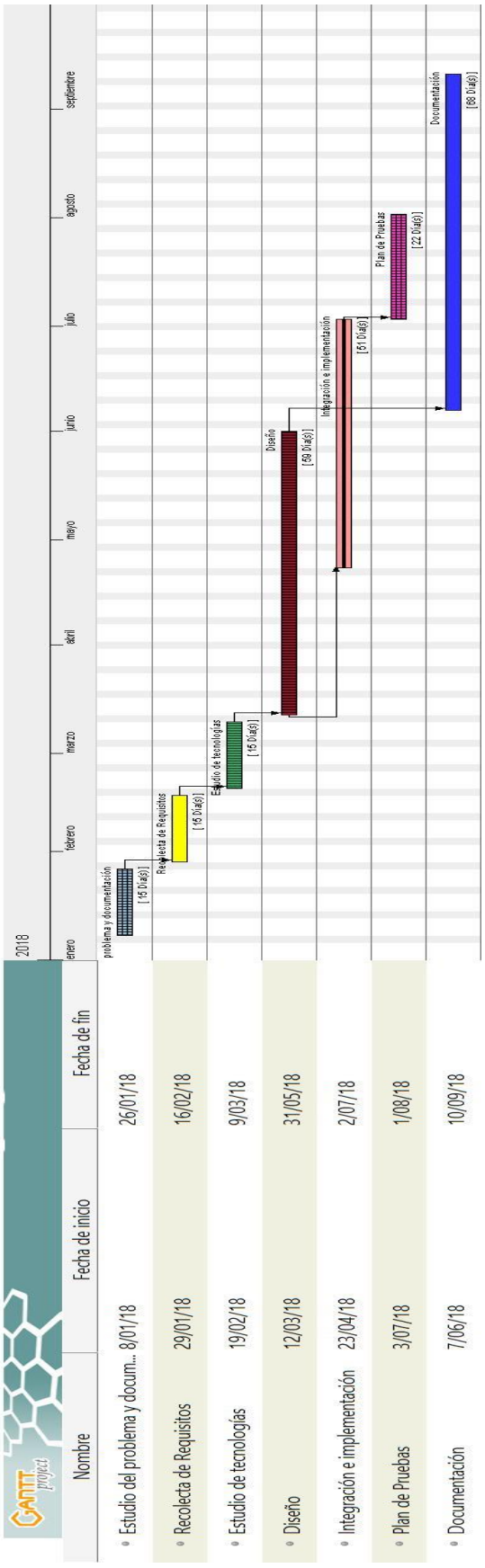

Figura 2.1: Figura de diagrama de Gantt

2.2.- Presupuesto

En este apartado se calculará y desglosará el presupuesto necesario para la elaboración del proyecto. Con este objetivo, se considerarán tanto los costes de personal, como el coste material y otros costes más que han logrado sacar adelante este proyecto.

2.2.1.- Coste de personal

Los honorarios corresponderán a los del Graduado en Ingeniería en Tecnologías de las Telecomunicaciones, puesto que ha sido el encargado del proyecto. La elaboración de este proyecto ha llevado 8 meses, tomando en cuenta 20 días laborables al mes, pese a que también ha existido desarrollo incluso en días no laborables. Con estas cifras, se obtiene un total de 160 días de elaboración. Se tendrá en cuenta también que no se ha realizado una jornada laboral de ocho horas diarias, sino que el proyecto suma un número de horas equivalente a 400 horas.

A su vez, como responsable de supervisar el proyecto, se tendrán en cuenta las horas de trabajo realizadas por el tutor de este proyecto, cuyos honorarios corresponderán a los de un Ingeniero Superior de Telecomunicaciones. Entre todas las reuniones planificadas y la supervisión on-line del trabajo, se han considerado la suma de 10 días laborables con un horario de 8 horas laborables, lo que conforman 80 horas trabajadas.

Con ambos cargos establecidos, se deben valorar los salarios mensuales de cada uno de ellos. En la siguiente figura, se presentan los costes que se consideran de personal (con las cargas sociales añadidas):

Personal	Cargo	Salario mensual	Salario por hora	Tiempo	Coste Total [€]
Iván Benítez Martín	Graduado en Ingeniería en Tecnologías de las Telecomunicaciones	31.000€	16,14 €/h	400 h	8392,80 €
Jesús Arias Fisteus	Ingeniero superior de Telecomunicaciones	60.000€	31,25 €/h	30 h	1218,75 €

Tabla 2.1: Coste final del personal

2.2.2.- Coste de material

A continuación se presenta un listado de los materiales empleados en la elaboración de este proyecto:

- Ordenador personal HP Pavilion con sistema operativo Windows 10, valorado en 1.000 €.
- Huawei P8 Lite, valorado en el momento de su compra en 400 €.

Por esta parte, es posible calcular su amortización si se suponen una vida útil de 3 años para los dispositivos móviles, y una vida útil de 5 para los ordenadores portátiles. Además, los cortes imputables de los materiales se pueden extraer de la siguiente fórmula:

$$Coste\ imputable\ [€] = \frac{Número\ de\ meses\ de\ uso}{Vida\ útil} * Coste\ total\ del\ equipo$$

Ecuación 2.1: Fórmula de coste imputable de materiales

En la Ecuación 1 no se considera el porcentaje de uso del equipo dentro del proyecto ya que ha sido del 100 %. Así, en la siguiente tabla se desglosan los costes imputables de los materiales:

Equipo	Coste (€)	Meses de uso	Vida útil (meses)	Coste imputable (€)
Smartphone Huawei P8 Lite	400	8	48	66,67
HP Pavilion	1000	8	48	166,67
TOTAL				233,33 €

Tabla 2.2: Coste final del material

2.2.3.- Coste de Software y Servicios en la Nube

En este apartado, se observan los costes relacionados con el software y con los servicios de Azure contratados.

Azure es un servicio bajo demanda que acepta el “Pago por uso”, es decir, que según las infraestructuras de Microsoft que se utilicen, la factura mensual variará. Por lo tanto, no solo se tendrá que tener en cuenta el coste de estas infraestructuras, si no el mantenimiento de estas por parte del equipo técnico. A continuación, en la Tabla 2.4 se mostrarán los costes mensuales de este servicio en la nube:

Mes	Precio
Marzo 2018	11,21
Abril 2018	10,2
Junio 2018	9,5
Julio 2018	9,97
Agosto 2018	9,93
Septiembre 2018	9,95
Total	60,76

Tabla 2.3: Coste del servicio en la nube

Por otro lado, pese a que, al tratarse de un proyecto de tipo académico, ha sido posible el uso del software gratuito que proporciona la Universidad Carlos III a sus alumnos, es necesario realizar también un desglose de costes de licencias de software.

Software	Coste (€)	Meses de uso	Vida útil (meses)	Coste imputable (€)
Android Studio	0	8	-	0
Licencia de Windows 10	0	8	-	0
Licencia de Microsoft Office	99,99 €	8	9	88,88
TOTAL				88,88

Tabla 2.4: Coste de software con amortización

2.2.4.- Coste final

Finalmente, tras el desglose de los costes más importantes, a continuación en la Tabla 2.5 se muestra el coste final del proyecto:

Motivo del coste	Coste
Coste de Personal	9611,55 €
Coste de material	233,33 €
Software con amortización	88,88 €
Coste del servicio en la nube	60,76 €
Costes indirectos (20% de todo lo anterior)	1998,90 €
IVA (30% de todo lo anterior)	3598,03 €
TOTAL	15591,46 €

Tabla 2.5: Coste final del proyecto

Se recuerda que, al considerarse este proyecto de desarrollo, no se han tenido en cuenta los costes de gestión ni presupuestado las tareas de mantenimiento, ya que, en teoría, iniciarían al fin del presente proyecto.

Capítulo 3: Estado del arte

3.1.- Machine Learning y sistemas de recomendación de películas

Lo que se dispone a continuación es un análisis descriptivo de diversas características de esta no tan nueva tecnología llamada "aprendizaje en máquina" o comúnmente más conocido, "Machine Learning": ¿en qué consiste? ¿Cómo funciona? ¿A qué podemos aplicar las soluciones que aporta esta nueva ciencia?

De una forma algo simplificada, Samuel Arthur, el padre del "Machine Learning" (a partir de este momento será denominado ML), definía esta disciplina científica como: "un campo de estudio que otorga a las computadoras la habilidad de aprender sin ser explícitamente programadas" (Mund, 2015)[4]. Definiéndolo de una manera más técnica, ML explora la construcción de algoritmos estudiados a raíz de una serie de muestras de datos entrenadas para predecir soluciones en múltiples aplicaciones, sean sistemas de búsqueda, asistentes digitales o coches autónomos modernos, entre otras. Y ese tipo de aplicaciones y las innumerables ventajas que aporta esta ciencia analítica, están dirigiendo el camino de la

[4] Mund, S. (2015). *Microsoft Azure machine learning*. Packt Publishing.

innovación y de la industria a la búsqueda de modelos predictivos más complejos. Ahora más que nunca, el fuerte poder de computación mediante algoritmos complejos y acceso a cantidades masivas de datos que proporciona el ML es una herramienta apreciadísima que otorga oportunidades únicas en el campo de la tecnología.

Un ejemplo de esta tecnología ilustrará mejor su importancia. Una consultora de Machine Learning, llamada Kaggle[5], organizó una competición para lograr el algoritmo que mejor clasificara imágenes con respecto a si aparecía un perro o un gato en ellas. Para ello, se tenían unas muestras de entrenamiento que ascendían a 25,000 imágenes etiquetadas. Para entregar la solución final, el algoritmo tendría que ser capaz de etiquetar 12,500 imágenes no clasificadas. Durante la competición, los usuarios barajaron distintos métodos para esa clasificación, como el estudio de formas, colores o texturas, lo que les permitía el aprendizaje de ejemplos (Brink, Richards & Fetherolf, 2017)[6]. En Machine Learning, estas estrategias se construyen sobre complejos algoritmos desarrollados en diferentes disciplinas en las que todas tienen en común exactamente eso, el aprendizaje de los ejemplos o experiencia, y la habilidad de aplicarlo o generalizarlo. Así, después de las millones de iteraciones, en las cuales el algoritmo realizaba la clasificación (con los datos de entrenamiento), comparaba los resultados y los ajustaba, incrementó la mejora de este. El algoritmo ganador llegó a ser capaz de clasificar el 98.91% de las imágenes no clasificadas correctamente. Y, ¿cuál sería el ratio de error humano? El 7%. Este ejemplo describe muy bien el "Aprendizaje en Máquina Supervisado".

Podemos clasificar el aprendizaje en máquina en diversos tipos según sus problemas. Así, si el objetivo es, por ejemplo, predecir nuevos casos de estudio, nuestro aprendizaje será un "**Aprendizaje en Máquina Supervisado**". En este tipo de aprendizaje, el algoritmo analizará unas muestras de datos que le permitirán ser entrenado para llegar a esta solución. En el siguiente esquema, se ilustra este mecanismo:

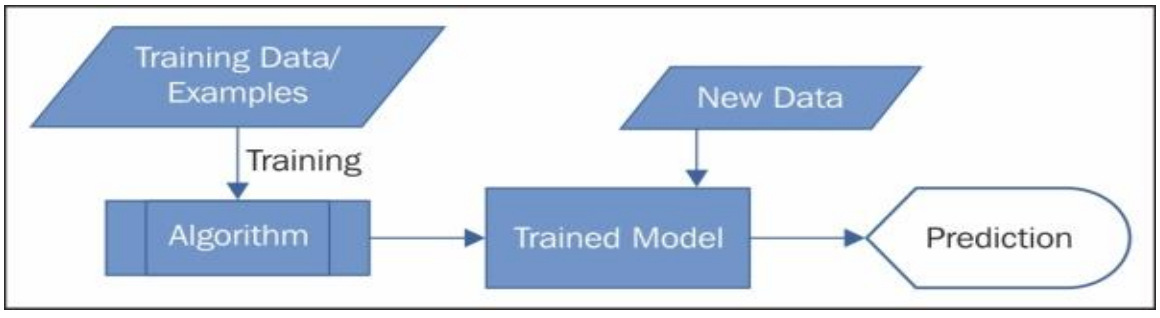

Figura 3.1: Esquema de ML[7]

[5] Visite "Dogs vs. Cats" en www.kaggle.com/c/dogs-vs-cats. Dogs vs. Cats | Kaggle. (2018). Recogido de https://www.kaggle.com/c/dogs-vs-cats

[6] Brink, H., Richards, J., & Fetherolf, M. (2017). *Real-world machine learning* (p. Chapter 1: What is Machine Learning). Shelter Island: Manning.

[7] Ilustración recogida de: Mund, S. (2015). *Microsoft Azure machine learning*. Packt Publishing.

En el caso en el que no existiera el objetivo de entrenar una serie de datos entrenados, sino de agruparlos, se hablaría de un "**Aprendizaje en máquina no supervisado**". Por ejemplo, en una campaña electoral se quiere estudiar una muestra de población en referencia a la intención de voto. Así, una muestra de datos con las características de la población puede ser ajustada por una serie de algoritmos genéricos que agrupen los datos en unos clústeres definidos.

Además de los dos mencionados, existen más tipos de aprendizaje en máquina. Sin embargo, el tipo de sistema que se estudiará a lo largo de este proyecto será el de **sistemas de recomendación**. Estos sistemas son muy comunes actualmente y se utilizan, entre otros, en sitios web como Amazon, que muestra productos en venta basados en las características del consumidor o, incluso, en su historial de navegación. Así, un sistema de recomendación está siempre personalizado a partir de unos datos preparados.

A la hora de construir un sistema de recomendación, es posible clasificar su funcionalidad y dificultad respecto a los campos de las muestras de datos en que se fije el estudio.

En los sistemas de recomendación **colaborativa**, el estudio se basará en la información que ofrece el comportamiento anterior o las opiniones de distintos usuarios de la comunidad que ofrece los datos, ofreciendo la predicción que probablemente más guste. Durante años se han explorado e investigado sus ventajas, como han hecho Jannac, Zanker y Felfernig (2010)[8], sus limitaciones y funcionalidad, convirtiéndolo en un sistema exitoso dentro de la ciencia del estudio de datos. Dada una base de datos y el identificador del usuario actual al que se le predecirá el ítem, se identificarán los usuarios con las preferencias más similares a este usuario activo, y por cada ítem que dicho usuario no haya reproducido, se calculará una predicción basada en la evaluación que esos usuarios similares hayan hecho del ítem. Sin embargo, algunas de las desventajas es que se asume que, si en el pasado esos usuarios han tenido gustos parecidos, los continuarán teniendo en el futuro, cuando perfectamente un usuario puede haber perdido el interés que en el pasado tuvo por ese ítem.

Al observar el tipo de sistemas de recomendación anterior, se puede llegar a la conclusión de que no hace falta saber nada del contenido de los ítems para realizar una recomendación. Así además se evita el coste de proveer descripciones actualizadas del contenido de los ítems. Sin embargo, si se pusiera un ejemplo común en el mundo real: si al usuario A le encantan los libros de ciencia ficción, muy posiblemente le guste *Fundación* de Isaac Asimov, un libro de ciencia ficción. Para llevar a cabo la tarea de esta recomendación, no se necesitaría una larga comunidad de usuarios, únicamente una descripción del contenido de los ítems y un perfil del usuario al que se le va a recomendar. Este tipo de sistemas de

[8] Jannach, D., Zanker, M., Felfernig, A., & Friedrich, G. (2010). *Recommender Systems: An Introduction*. Cambridge: Cambridge University Press. doi:10.1017/CBO9780511763113

recomendación, reciben el nombre de **sistemas de recomendación basados en contenidos**.

Muchas técnicas de ML han aplicado este tipo de sistemas con árboles de decisión, redes neuronales o clasificadores bayesianos. Por ejemplo, utilizando el clasificador Bayesiano (Lew, Sebe, Djeraba & Jain, 2006)[9], se han determinado métodos para estimar la probabilidad de que un ítem pertenezca a una clase C_i, como la siguiente función de probabilidad:

$$P(X) = \prod_{k=1}^{n} P(x_k|C_i),$$

donde X, cada instancia del ítem, es descrita como conjunción de los atributos del ítem $<x_1, x_2, \ldots x_k>$. Sin embargo, la mayor desventaja de este sistema de recomendación es que, si el contenido no provee suficiente información para clasificar esos ítems, la recomendación será imposible. Se entiende entonces que una asociación de la metodología de varios sistemas podría convertirse en un sistema de recomendación óptimo (Son & Kim, 2017)[10].

Actualmente, uno de los sistemas de recomendación más potentes que existe es el sistema de recomendación de **Netflix**. La revista tecnológica *Wired*, (Plummer, 2017)[11], explicaba qué se esconde ante uno de los mayores secretos de la empresa. Compara el sistema de recomendación con una silla de tres patas.

La "primera pata" constituye el historial de reproducciones del usuario: cuándo lo ha visto, lo que ha visto después, antes y en qué momento del día lo ha visto. La "segunda pata" se basa en una serie de etiquetas que contienen todas las series que se encuentran en la plataforma. No son tags de evaluaciones, sino más especializados: "*The tags they use range massively from how cerebral the piece is, to whether it has an ensemble cast, is set in space, or stars a corrupt cop.*"

Entre la primera y la segunda pata entra a jugar el algoritmo de "Machine Learning", para evaluar la importancia del historial, si el usuario vio 10 minutos y luego cambió de serie, lo que vio hace un mes. Con este tipo de datos, se mide mejor los sentimientos del usuario y en qué momento recomendar o no una serie según su historial. Lo que genera son como "comunidades de gustos" o así lo llama el escritor, creando grupos con los mismos gustos que el usuario.

Las etiquetas son universales. Sin embargo, hay un grupo de sub-tags que se alimentan de las características de usuarios de distintos países con su propio

[9] Lew, M., Sebe, N., Djeraba, C., & Jain, R. (2006). Content-based multimedia information retrieval. *ACM Transactions On Multimedia Computing, Communications, And Applications*, *2*(1), 1-19. doi: 10.1145/1126004.1126005

[10] Son, J., & Kim, S. (2017). Content-based filtering for recommendation systems using multiattribute networks. *Expert Systems With Applications*, *89*, 404-412. doi: 10.1016/j.eswa.2017.08.008

[11] Plummer, L. (2017). This is how Netflix's top-secret recommendation system works. Retrieved from http://www.wired.co.uk/article/how-do-netflixs-algorithms-work-machine-learning-helps-to-predict-what-viewers-will-like

contexto cultural y lingüístico: "*For example, the word 'gritty' [as in, 'gritty drama'] may not translate into Spanish or French*".

Los datos con los que Netflix entrena a sus algoritmos pueden dividirse en implícitos y explícitos. Los datos explícitos son aquellos que el usuario provee al sistema, por ejemplo, si indica que le ha gustado la serie *The Alienist*. Datos implícitos son aquellos datos que recoge el sistema del comportamiento del usuario, es decir, si ha reproducido en su casa la serie *The Crown* y la ha finalizado en un maratón. Los analistas de la compañía consideran este último tipo de datos, los implícitos, los más útiles y valiosos para el éxito de su sistema.

La cantidad de datos que maneja Netflix se aleja mucho de los recursos que este estudio posee a la hora de realizar un sistema de recomendación. Sin embargo, sí que puede sugerir que conviene evitar los sistemas de recomendación "puros". Así, se llega al tipo de sistema que se ha utilizado en este proyecto, los **sistemas de recomendación híbridos**.

El sistema de recomendación híbrido basa su implementación en la combinación de distintos sistemas de recomendación, elegidos según las necesidades del problema a resolver. Suresh Kumar Gorakala, en su libro *Building Recommendations Engines*[12], enumeraba algunas de sus ventajas, entre las que se encuentran una mayor robustez y escalabilidad que cualquiera de los modelos individuales y una mayor precisión debido a las mencionadas combinaciones.

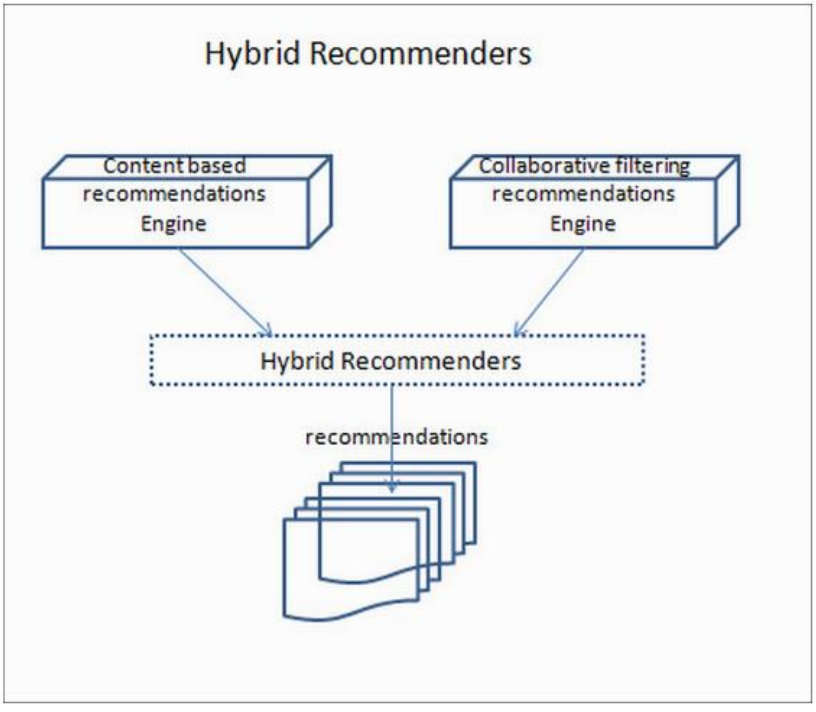

Figura 3.2: *Feature combination method*[13]

[12] Gorakala, S. (2016). *Building Recommendation Engines*. Packt Publishing. ISBN: 978-1-78588-353-8
[13] Ilustración obtenida de: Gorakala, S. (2016). *Building Recommendation Engines*. Packt Publishing. ISBN: 978-1-78588-353-8

A la hora de construir estos sistemas, los métodos más comunes son:

- Métodos ponderados, en los que las recomendaciones finales serían resultado de la combinación, mayormente lineal, de los resultados de todos los sistemas de recomendación disponibles. Durante la primera fase del desarrollo del sistema se otorgan pesos iguales a cada uno de los resultados disponibles y, gradualmente, estos pesos serán ponderados en función de las respuestas de los usuarios a esas recomendaciones.
- Los métodos mixtos son mayormente utilizados en aquellos sistemas que sufren de escasez de datos. Así, las recomendaciones finales son generadas de manera independiente y finalmente mezcladas antes de entregárselas al usuario.
- El método de cascada utiliza de manera secuencial los sistemas de recomendación basados en contenido y los colaborativos. Así, mientras que las recomendaciones son generadas mediante el filtro colaborativo, el sistema basado en contenido organiza estos resultados, mostrando como resultado una serie de recomendaciones o listas clasificadas,
- El método utilizado en este estudio, el método de combinación de características (en inglés, *Feature Combination Method*), que, como el método de cascada, también combina los sistemas de recomendación basados en contenido junto a los colaborativos. Así, del sistema basado en contenido extraemos las propiedades y características del contenido del ítem relacionándolas con la información que nos otorga el comportamiento anterior del usuario que ha valorado el ítem.

3.2- Aplicaciones Android

En 2005, momento en el que se sucede la compra de Android por parte de la multimillonaria Google, nadie sabía nada acerca de la empresa o de lo que Google tenía pensado hacer con ella. No fue hasta 2007, año en el que Google anuncia la primera plataforma abierta para móviles del mundo.

A partir de ese momento, cualquier desarrollador de software podría lanzar aplicaciones dirigidas a varios dispositivos sin tener que cambiar muchas líneas de código base. Y así, con esa filosofía de “open source”, se construirían muchas de las aplicaciones que hoy en día se siguen utilizando. Esta filosofía nace de la llamada licencia Apache, que prácticamente garantiza libertad para usar el software para cualquier propósito[14].

Esta tecnología se compone de varios componentes que permiten tanto a desarrolladores como fabricantes de dispositivos trabajar independientemente, y

[14] Para más información, visite http://www.apache.org/licenses/LICENSE-2.0.

se puede descomponer en cinco piezas básicas, que se disponen en la siguiente figura:

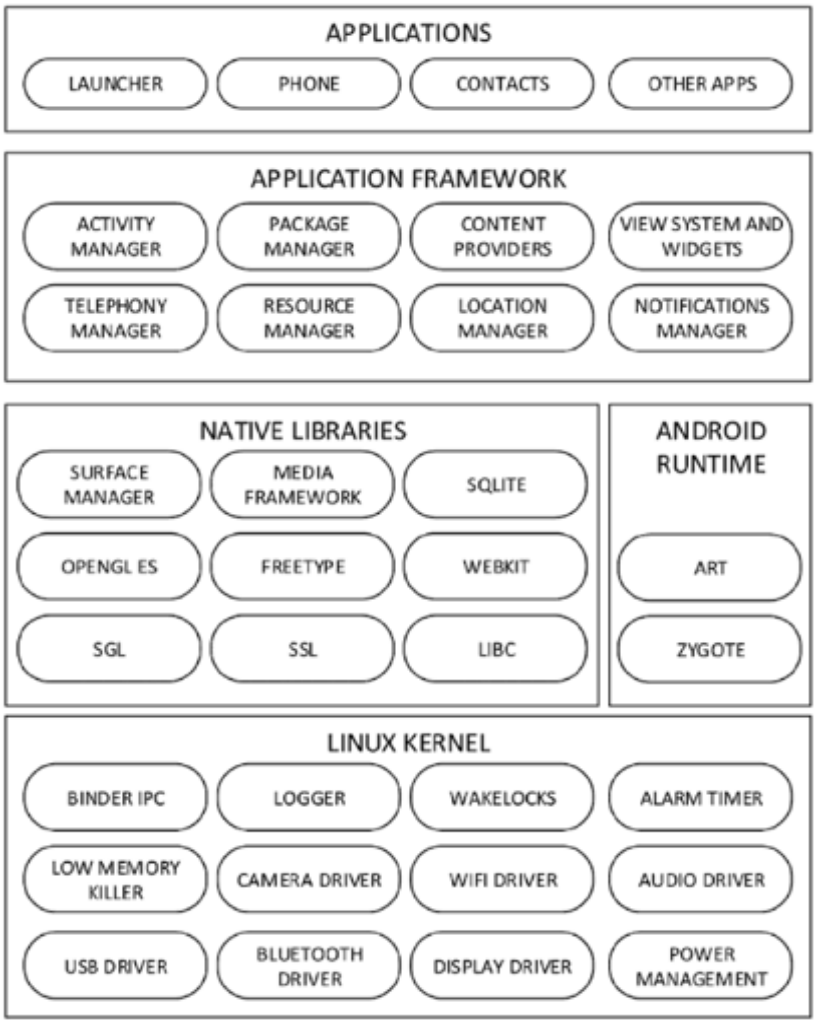

Figura 3.3: Arquitectura de Android[15]

En el primer apartado, se disponen las **aplicaciones del teléfono** que se encuentran al más alto nivel. Son robustas aplicaciones que soportan todas las necesidades del dispositivo, como llamadas, mensajería, Internet y demás terceras aplicaciones. Ya que Java es mundialmente conocida, el jefe ejecutivo de Android, Andy Rubin, apostó por este lenguaje para desarrollar estas aplicaciones (Krajci & Cummings, 2013)[16].

La capa de **estructura de aplicaciones** provee a los desarrolladores las herramientas para trabajar en aplicaciones del sistema mediante clases java y librerías entre las que se incluyen las que aparecen en la figura 3.3, como la "*Activity Manager*", que gestiona el ciclo de vida activo de las aplicaciones, el "*Content Provider*", que gestiona los datos que comparten varias aplicaciones, etc.

[15] Imagen obtenida de: Cinar, O. (2015). *Android Quick APIs Reference* (p. 2). Berkeley, CA: Apress.
[16] Krajci, I., & Cummings, D. (2013). *Android on x86* (pp. 1-8). [New York, NY]: ApressOpen. doi: 10.1007/978-1-4302-6131-5

En el caso de las **librerías nativas** de Android, el camino se separa. Tanto las librerías como Android Runtime existen en el mismo espacio. Las librerías guían al dispositivo mediante una serie de instrucciones, compiladas y preinstaladas en binarios C/C++. Entre sus funciones se encuentran:

- La base de datos que utiliza Android para guardar información entre las sesiones de un dispositivo Android es SQLite, que se guarda en la memoria interna del dispositivo.
- WebKit es el navegador web por defecto en los sistemas Android y permite una rápida y eficiente disposición de archivos HTML.
- El procesador de gráficos en Android es el motor OpenGL, que permite renderizar gráficos 2D y 3D.

En la capa de **Android Runtime**, se disponen tanto el núcleo de las librerías Java como la máquina virtual *Dalvik*, que no es más que la implementación de Google de Java, optimizada para dispositivos móviles.

Por último, de la capa más baja se ocupa el **Kernel de Linux**, que otorga el acceso más directo al hardware posible, por lo que los drivers son escritos en el espacio del kernel, actuando de la manera más eficiente posible y entre sus tareas están manejar la potencia y la carga de batería, operar con la pantalla del dispositivo, etc. Además, también es un proyecto de código abierto.

Desde la compra de Android por parte de Google, el cargo de desarrollo y mantenimiento del software de Android lo ocupa el Android Open Source Project. Una de las funcionalidades que conserva su desarrollo es la de que nuevos dispositivos puedan ejecutar aplicaciones desarrolladas en versiones anteriores de Android (exactamente hasta Cupcake 1.5). Con cada nueva SDK disponible a los desarrolladores, nuevas características son añadidas, por lo que siempre se debe tener en mente las características de versiones pasadas cuando se desarrollen nuevas aplicaciones. A continuación, un repaso de las versiones de Android y en la figura 3.4 la distribución de uso de las APIs a fecha de febrero de 2018:

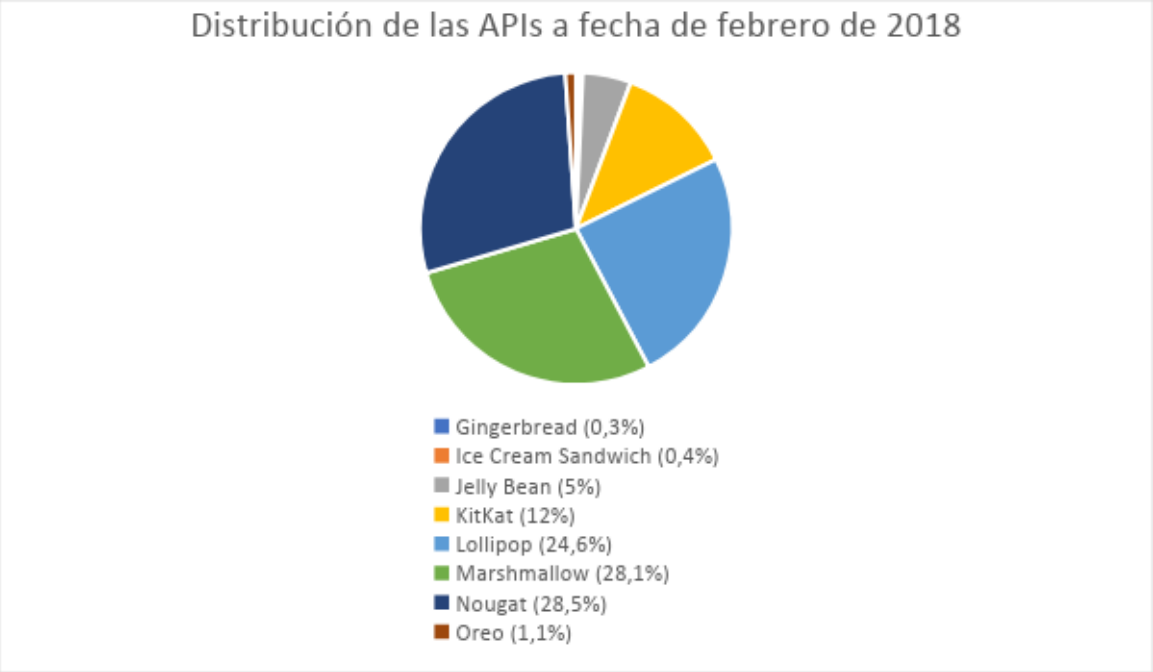

Figura 3.4: Distribución de APIs

- **Astro (1.0)** se convierte en el punto de partida de la compañía. Introducida al público en septiembre de 2008 en el HTC Dream, Astro incluía muchas aplicaciones que, hoy en día, siguen vigentes: Android market, navegador web, Gmail, Google Maps, mensajería, Youtube, etc.
- **Cupcake (1.5)** fue la catapulta y la primera versión con nombre de postre (esta tendencia sigue vigente). Estrenada en abril de 2009, introdujo cambios claves de la anterior versión, como soporte para teclado virtual y widgets, soporte estéreo para dispositivos con tecnología Bluetooth.
- En **Donut (1.6)**, Android actualizaba el kernel de Linux y así lo haría en todas las versiones, así como mejoras en búsquedas, soporte para pantallas WVGA y funcionalidad y velocidad de la cámara.
- **Éclair (2.0/2.1)** continuaba mejorando la velocidad en diferentes aplicaciones. El motivo de que haya dos versiones distintas de la propia Éclair, es que se introdujo una segunda versión que corregía pequeños errores y actualizaciones en la API.
- En mayo de 2010, **Froyo (2.2x)** sale al mercado, siendo Nexus One de Google el primer dispositivo el mercado con esta SDK integrada. Se incluía soporte a Adobe Flash, funcionalidad hotspot de Wi-Fi y mejoras de optimización.
- Nexus S de Google introdujo por primera vez en diciembre de 2010 el SDK **Gingerbread (2.3x)** y bajo el brazo traía soporte para WxGA y pantallas grandes, soporte para más sensores internos, como

giroscopios y barómetros, soporte para varias cámaras, mejoras en el teclado virtual y lectura de etiquetas NFC.

- La primera versión creada específicamente para tablets es **Honeycomb (3.x)**, que introducía un rediseño del teclado de pantalla, múltiples pestañas de navegación para facilitar el uso y soporte para procesadores con múltiples núcleos.
- **Ice Cream Sandwich (4.0.x)** sale al mercado en octubre de 2011 con el kernel de Linux 3.0.1. El primer dispositivo en llevarlo es el Samsung Galaxy Nexus que tiene una gran acogida en el mercado tecnológico, ya que esta versión contiene múltiples mejoras de la interfaz de usuario de Android, como reconocimiento facial para desbloqueo, aceleración del hardware de esta IU y botones de software introducidos ya en Honeycomb.
- En julio de 2012 y septiembre de 2013**, Jelly Bean (4.1.x) y KitKat (4.4)** continúan el legado de la anterior versión, centrando sus mejoras en la experiencia de usuario como soporte para mejores gráficos de juego y audio multicanal (Jelly Bean) o capacidad de impresión sin cables y ART, una nueva máquina virtual que sustituiría a Dalvik (KitKat).
- **Lollipop (5.0)** introduce por primera vez en noviembre de 2014 el Material Design que permite un diseño con una sensible interfaz de usuario, causando que el movimiento de respuesta sea más natural y permite mayor facilidad de navegación, además de sus colores vivos. Características de ahorro de batería y llamadas de voz de alta definición son otras de las mejoras que presenta esta versión.
- El soporte para huellas dactilares, como característica más novedosa, llega con la versión **Marshmallow (6.0)**, lanzada en octubre de 2015. Además, se incluye el sistema de pago por móvil Android Pay además de mejoras en rapidez y estabilidad.
- **Nougat (7.0)** o "turrón" en español aparece en agosto de 2016. En ella, se mejoran las animaciones y se incorpora JAVA 8. Comienza a utilizar Google Chrome como WebView, lo que ahorra consumo en RAM y, además, las instalaciones incrementan en velocidad gracias a que se compilan durante su ejecución.
- La última versión hasta la fecha de escritura de este proyecto es **Oreo (8.0)**, lanzada en agosto de 2017 y entre sus novedades se encuentran con Wi-Fi Aware, lo que significa que los dispositivos compatibles con reconocimiento de Wi-Fi podrán detectar y comunicarse con dispositivos cercanos a través de Wi-Fi sin necesidad de tener punto de acceso por Internet.

3.3- Microsoft Azure Machine Learning

La computación en la nube o "cloud computing" nace en su origen de la capacidad de monetizar computaciones complejas asociadas a operaciones IT. Una definición más completa la ofrece el NIST (National Institute of Standards and Technologies), que define "Cloud Computing" como: "Un modelo que permite conveniente y bajo demanda un acceso de red a un grupo compartido de recursos informáticos configurables (por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios) que se pueden aprovisionar y liberar rápidamente con un mínimo esfuerzo de gestión o interacción del proveedor de servicios"[17]. La nube, a su vez, es una red de centros de datos, cada uno compuesto por varios miles de computadoras trabajando de una manera conjunta con el objetivo de realizar las funciones de software ofreciendo acceso por parte de los usuarios a potentes plataformas, aplicaciones y servicios [18] . Pero "Cloud Computing" no implica solo una nube. El término de "nube" simboliza internet como red de redes. Entre las características más destacadas de este servicio, se pueden encontrar:

1. **Escalable**, ya que ofrece ayuda de una manera dinámica para evitar escenarios de cuellos de botella.
2. Los usuarios no se preocupan por el **mantenimiento** y actualización de recursos, puesto que el proveedor de servicios se encarga de ello.
3. Los recursos se comparten en un grupo de **acceso compartido** de usuarios.
4. No importa desde donde accedas al servicio, puesto que tiene **independencia de localización**.
5. El servicio se caracteriza por su **agilidad**, puesto que ayuda de una manera rápida y sin coste a un reaprovisionamiento de recursos.

[17] Anderson J., University E., & Rainie L. (2005). Technical seminar report on cloud computing. *Intel Executive Summary*.
[18] Wang L., Ranjan R., Chen J. & Benatallah B. (2017). *Cloud Computing: Methodology, Systems and Applications*. CRC Press. ISBN: 978-1-351-83309-7

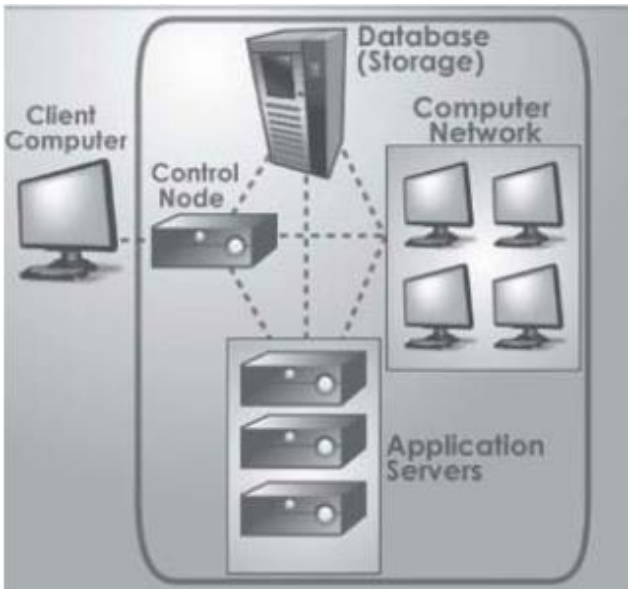

Figura 3.5: Arquitectura básica de "Cloud Computing"[19]

Es por ello por lo que Microsoft presentó en 2008 un servicio en la nube alojado en los Data Centers de Microsoft, llamado en su origen Windows Azure u posteriormente Microsoft Azure. Pese a que es un servicio en la nube medianamente novedoso, Azure ha evolucionado a pasos agigantados desde que proveía software como servicio (SaaS) en su plataforma anterior, Office 365, en la que permitía a los usuarios acceder a todas las herramientas y productos de Microsoft sin tener que implementar una infraestructura local significativa.

Aunque se ha identificado a Microsoft Office 365 como un servicio SaaS, existen dos tipos más de servicios en la nube: infraestructura como servicio (IaaS) y plataforma como servicio (PaaS). En Azure, es fácil reconocer los diferentes tipos de servicios. Por ejemplo, a Office 365 se le provee poder computacional por parte de Azure, desde Active Directory, e incluso Azure es mayormente reconocido por su IaaS, ya que incluye máquinas y redes virtuales, soluciones de almacenamiento y servicios de recuperación. También se pueden encontrar distintos servicios PaaS como Azure Mobile Services, Azure SQL Database, Azure Websites, Azure Content Delivery Network (CDN) o Azure BizTalk Services.[20]

Microsoft Azure Machine Learning (a continuación, se denominará Azure ML) es un servicio en la nube accesible desde cualquier navegador a Internet, en el que se dispone un entorno desarrollado y operacional que posibilita la construcción de sistemas basados en la tecnología ML. Permite construir, entrenar, testear y establecer análisis predictivos en cualquiera de sus aplicaciones. Este tipo de sistemas beneficia al desarrollador a la hora de manejar cantidades masivas de

[19] Ilustración obtenida de: Wang L., Ranjan R., Chen J. & Benatallah B. (2017). *Cloud Computing: Methodology, Systems and Applications*. CRC Press. ISBN: 978-1-351-83309-7

[20] Copeland, M., Soh, J., Puca, A., Manning, M., & Gollob, D. (2015). *Microsoft Azure: Planning, Deploying, and Managing Your Data Center in the Cloud*. Berkeley, CA: Apress.

datos, ya que, al construirse en la nube, se integra con softwares relacionados al Big Data que procesan servicios Azure como Hadoop. Un ejemplo de cómo se dispone el área de trabajo de este servicio se encuentra en la Figura 3.6:

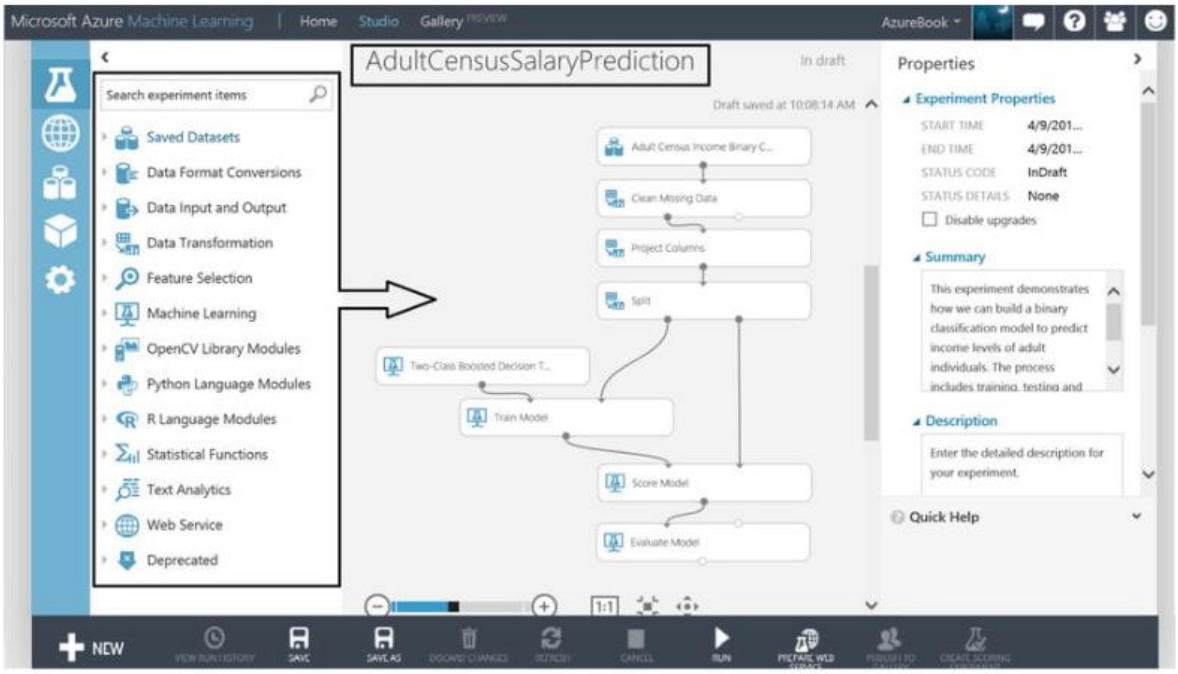

Figura 3.6: Experimento en Machine Learning Studio[21]

3.4.- Aplicaciones como competencia

3.4.1.- TMDb

En 2008, una comunidad de desarrolladores comenzó a construir una inmensa base de datos en torno a películas y series de televisión. Su objetivo era equipararse a la archiconocida IMDb (Internet Movie Database), pero enfocándolo en torno al uso libre y sin ánimo de lucro de estos datos. Así comenzó The Movie Database o TMDb.

Además, también funciona como red social. Es posible registrarte e ingresar dentro de la comunidad para poder valorar aquellas películas y series de televisión existentes en la aplicación.

[21] Ilustración obtenida de: Copeland, M., Soh, J., Puca, A., Manning, M., & Gollob, D. (2015). *Microsoft Azure: Planning, Deploying, and Managing Your Data Center in the Cloud*. Berkeley, CA: Apress.

Figura 3.7: Fragmento del servicio web

Pese a no ser una base de datos tan completa como IMDb, multitud de aplicaciones hacen uso de su API, como Kodi, MythTV o Plex. La API de TMDb ofrece una manera rápida, consistente y de confianza de conseguir los datos que se requieren. Para ello, se ofrece la oportunidad desde la página web de adquirir la clave para esta API de dos maneras distintas dependiendo de la finalidad de la aplicación a la que la quieres integrar: una clave de API comercial, si estás interesado en sacar beneficios de la aplicación o una clave de API para desarrolladores, sin beneficios.

Pese a que todos sus métodos están disponibles en su página web[22], a continuación, se presentan tres métodos sencillos para encontrar películas, series de televisión o actores en TMDb:

- /search. Se inserta una búsqueda mediante un *string* y devuelve los resultados más cercanos, ya sea por título original, traducido a cualquier idioma, etc.
- /discover. Este método es útil si en vez de realizar una búsqueda al uso, se requiere de filtros o valores definidos para descubrir un conjunto de resultados.
- /find. En el caso en que ya se disponga de un servicio web y se quiera añadir esta herramienta a la aplicación, este método permite encontrar resultados a partir de un identificador externo como, por ejemplo, un identificador de IMDb.

[22] Más información en developers.themoviedb.org

3.4.2.- FilmAffinity

En 2002, un crítico de cine llamado Pablo Kurt Verdú Schumann y un programador Daniel Nicolás unieron fuerzas para desarrollar un sistema de recomendación llamada "Almas Gemelas", el cual aparejaba usuarios afines en cuanto a las puntuaciones que daban a las películas. Más detalladamente, el algoritmo compara las puntuaciones del usuario con el de todas las personas registradas, que, hoy en día, se acerca a un millón de usuarios, y su alma gemela será aquella que más puntuaciones iguales a el usuario tenga.

Poco a poco, fueron integrando más ítems en su base de datos, además de la posibilidad de realizar críticas de las películas mismas. Esto derivó en lo que se conoce hoy en día como FilmAffinity. Actualmente, se considera una de las redes sociales más importantes de cine, en cuyos foros se encuentran multitud de discusiones de cine, entre otras interacciones entre usuarios.

Figura 3.8: Logotipo de FilmAffinity

3.5.- Servicios Web REST

Roy Fielding definió en su obra 'Architectural Styles and the Design of Network-based Software Architectures'[23] un nuevo enfoque de desarrollo de proyectos y servicios web, API REST (Representational State Transfer). En este nuevo enfoque, Fielding desarrollaría una familia de arquitecturas, así como un estándar, para obtener o manipular datos en cualquier formato, ya sea XML o JSON, utilizando HTTP para ello.

Su origen se basa en la necesidad de generar una alternativa a anteriores protocolos de intercambio de datos como es el caso de SOAP (Simple Object Access Protocol), considerado extraordinariamente complejo en algunas ocasiones y con la intención de proponer soluciones a casi cualquier comunicación necesaria. Una de las principales diferencias entre ambos servicios web es que SOAP es un protocolo orientado a RPC (Remote Procedure Call). Esto significa que a la hora de obtener datos del propio servicio, SOAP invoca métodos sobre un servicio remoto. En el caso de una API REST, se orienta el concepto a la accesibilidad de recursos, accesibles por identificadores sobre los que se podrán obtener o manipular recursos a través de distintos verbos HTTP.

[23] Fielding, R. (2000). *Architectural styles and the design of network-based software architectures*. California: University of California, Irvine.

¿Y cómo se pueden llevar a la práctica ambos conceptos? A continuación, se muestra un ejemplo que permita ilustrar las diferencias entre ambos servicios web.

En un servicio web clásico (SOAP), se tiene un servicio que nos proporciona accesibilidad a una librería de películas. Se puede observar así un método llamado GetAllMovies que, al invocarlo, se recogen todas las películas de esa filmoteca. Otro ejemplo sería intentar conseguir una película en especial al invocar el método GetMovieByTitle(), pasándole como argumento el título de la película.

Ahora sí, lo que se tendría en un servicio API REST es algo totalmente distinto. Como se pierde la idea de método y aparece la idea de recurso, lo que se invocará será una URI, por ejemplo, "/movies", que dará acceso al listado completo de películas. Eso será posible gracias a un verbo HTTP que indicará cuál es la operación que se desea hacer con este servicio. Así, conseguir el listado completo de películas sería posible con un verbo GET y una URI como la antes mencionada. Si se quisiera una película en especial, el verbo se mantendría constante, pero la URI cambiaría, añadiendo más información como, por ejemplo, "/movies/HarryPotter". Si a su vez, en vez de recoger ese recurso, se pretende añadir una película a esa filmoteca, sería el verbo HTTP lo que variaría, siendo PUT el indicado, por ejemplo, PUT y "/movies/Se7en".

3.6 Sistema de recomendación del módulo "Train Matchbox Recommender" de Azure

En el trabajo que realizaron en Microsoft[24], se comenzó asumiendo que el sistema de recomendación recibiría conjuntos de variables (x, y, Φ, r) correspondientes a las descripciones del usuario, las descripciones del ítem, otras características que describen el contexto de las valoraciones y las propias valoraciones, respectivamente. Se definió el vector de rasgo de usuario K-dimensional como s=Ux, donde U es una matriz Kxn de rasgos de usuario donde cada elemento u_{ki} es la contribución de la característica *i* a la dimensión de rasgo de usuario k. Además se define el vector de rasgo de ítem k-dimensional como t=Vy para una matriz de Kxm rasgo de ítem, V. Se modeló un sesgo, $b = \Phi^{\mathrm{T}} w$, donde w es el conjunto de pesos latente.

De esta manera las características denominadas con Φ serán la combinación entre las características de descripción de usuario x y las características de descripción de los ítems y, por lo que solo en estos casos se podría formular la siguiente fórmula:

[24] D. Stern, R. Herbrich y T. Graepel. *Matchbox: Large Scale Online Bayesian Recommendations.* Published in: *Proceedings of the 18th International World Wide Web Conference* 2009

$$b = x^{\mathrm{T}}u + y^{\mathrm{T}}v,$$

siendo u y v los sesgos de usuario e ítem.

Por lo tanto, se modela la valoración, r, como:

$$p(r|\mathbf{s},\mathbf{t},b) = \mathcal{N}(r|\mathbf{s}^{\top}\mathbf{t} + b, \beta^2)$$

,

donde β es la desviación estándar del ruido observado. Así, se adopta una forma bi-lineal en la cual la relación entre usuario e ítem viene dada por el producto interno del vector de rasgos del usuario con el vector de rasgos del ítem. La valoración esperada es proporcional a las longitudes de los vectores de rasgos s y t así como al coseno del ángulo entre ambos. Debido a requisitos de tiempo y memoria de los cálculos, los investigadores eligieron los límites del parámetro K como $K \ll n$ y $K \ll m$.

Factorización previa

A la hora de determinar cómo los usuarios y los ítems están mapeados al espacio de rango dimensional K y a w, el valor de las características de sesgo para las valoraciones, se ha de indicar los parámetros del modelo cuyo aprendizaje se desea: las variables U y V.

A la hora de hablar de la factorización previa, se debe tener en cuenta la reducción de los requisitos de memoria a dos parámetros (una media y una desviación estándar) por cada componente que permitirá una actuación más eficiente. De esta manera Stern, Herbrich y Graepel representaron sus creencias previas acerca de los valores de los antes mencionados parámetros del modelo mediante distribuciones gaussianas independientes en cada uno de los componentes de las matrices U y V y del vector w. Uno de los ejemplos de estas distribuciones Gaussianas:

$$p(\mathbf{U}) = \prod_{k=1}^{K}\prod_{i=1}^{n}\mathcal{N}(u_{ki};\mu_{ki},\sigma_{ki}^2).$$

Modelos de feedback y regresión ordinal

Una de las ventajas del enfoque descrito a lo largo de este capítulo es la capacidad de modelar diferentes tipos de valoración de usuarios-ítems. Para este modelo, se ha utilizado la regresión ordinal.

Un escenario muy común se da cuando los usuarios proporcionan un feedback sobre los ítems mediante una valoración numérica. En este sistema, el conjunto de datos relacionado con las valoraciones venía bajo un rango de cada película sobre cinco estrellas.

Así, la interpretación de cada usuario de esta escala puede ser diferente y el mapeo de de rango a valoración latente puede no ser lineal. Se asume que, para cada par de usuario-ítem, se observa una valoración $l \in 1, \dots, L$. Se relaciona la valoración latente, r, al rango *l* mediante un modelo de umbral acumulativo[25]. Para cada usuario, u, se mantienen umbrales específicos de usuario $b_u \in \mathbb{R}^{L-1}$, que organizan los ejes de valoración latentes en L intervalos consecutivos $(b_{u(i-1)}, b_{u(i)})$ de longitud variable y en cada intervalo se encuentra representada la región en la que el usuario otorga el mismo rango a un ítem. Así, se define un modelo generativo de una valoración como:

$$p(l = a|\mathbf{b}_u, r) = \begin{cases} \prod_{i=1}^{a-1} \mathbb{I}(r > \tilde{b}_{u(i-1)}) \prod_{i=a}^{L-1} \mathbb{I}(r < \tilde{b}_{u(i-1)}) & \text{if } 1 < a < L \\ \prod_{i=1}^{L-1} \mathbb{I}(r < \tilde{b}_{u(i-1)}) & \text{if } a = 1 \\ \prod_{i=1}^{L-1} \mathbb{I}(r > \tilde{b}_{u(i-1)}) & \text{if } a = L \end{cases}$$

donde

$$p(\tilde{b}_{ui}|b_{ui}, \tau) = \mathcal{N}(\tilde{b}_{ui}; b_{ui}, \tau^2)$$

y se posiciona un modelo gaussiano independiente previo en el umbral tal que:

$$p(b_{ui}) = \mathcal{N}(b_{ui}; \mu_i, \sigma_i^2)$$

Dinámicas

Un sistema de recomendación debe ser capaz de realizar un seguimiento a datos no estacionarios. Los gustos de los usuarios cambiarán con el tiempo y así, la popularidad de un ítem se incrementará o caerá según las tendencias. Además, si se utilizan los umbrales específicos de usuario antes mencionados, la escala de valoración individual de usuario puede también cambiar con el tiempo.

Este tipo de dinámicas han sido modeladas asumiendo que las variables latentes U, V, w y b cambian a lo largo del tiempo, mediante el uso de la suma de ruido Gaussiano en cada iteración. Por ilustrar esta explicación, se tiene:

$$p(b_l^{(t+1)}|b_l^{(t)}) = \mathcal{N}(b_l^{(t+1)}; b_l^{(t)}, \gamma^2)$$

donde t es un índice sobre pasos de tiempo, y en t_0, se usa el previo:

$$p(b_i^{(0)}) = \mathcal{N}(b_i; \mu_i, \sigma_i^2).$$

Por último, los tres autores británicos ilustraron un gráfico de factores para este modelo bilineal de valoraciones, que se indica a continuación:

[25] W. Chu and Z. Ghahramani. Gaussian processes for ordinal regression. Páginas 1019–1041, 2005.

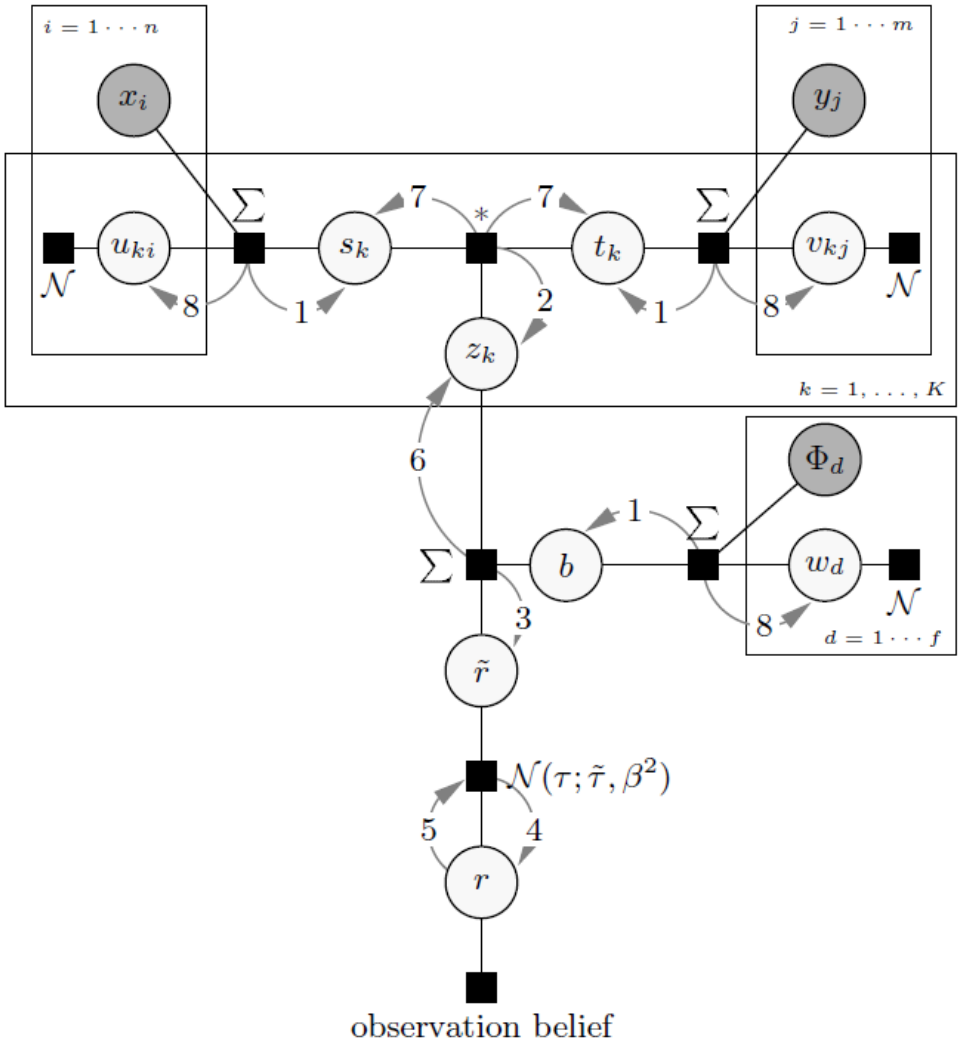

Figura 3.9: Gráfico de factores para el modelo de valoración bilineal[26]

Los rectángulos indican partes del gráfico que se repiten mediante repetición indexada por la variable en la esquina del rectángulo. Los factores etiquetados $\sum$ son factores de suma de la forma $\mathbb{I}[z = x + y]$. Las flechas numeradas corresponden a los mensajes. Mensajes con (1) son aquellos mensajes desde los que los factores de suma mapean desde las descripciones de usuario/ítem/sesgo hasta el espacio característico latente, $m_{\sum \to s_k}(s_k)$, $m_{\sum \to t_k}(t_k)$, y $m_{\sum \to b}(b)$. El mensaje con (2) es el mensaje $m_{\prod \to z_k}(z_k)$ del factor producto.

El mensaje con (3) corresponde al mensaje $m_{\sum \to \tilde{r}}(\tilde{r})$ desde el factor de suma total a la valoración latente $\tilde{r}$. Mensaje con (4) es el mensaje desde el factor de ruido Gaussiano a la valoración latente y ruidosa, r. Por último, el resto de mensajes son los mensajes revirtiéndose a los anteriores.

[26] Figura extraida de: D. Stern, R. Herbrich y T. Graepel. *Matchbox: Large Scale Online Bayesian Recommendations.* Published in: *Proceedings of the 18th International World Wide Web Conference* 2009

Capítulo 4: Requisitos

A lo largo de este capítulo se enumerarán los requisitos de software considerados para este proyecto. Este tipo de especificaciones describen el comportamiento del sistema desarrollado. Después de una larga investigación, se ha decidido utilizar una tabla de requisitos de software, que sirve como adaptación del formato de Especificación de Requisitos Software (ERS) según la última versión del estándar IEEE-830[27]. A continuación se presenta la tabla:

Identificador	RS-XX
Nombre	
Descripción	
Fuente	
Prioridad	
Estabilidad	
Prerrequisitos	
Necesidad	
Verificabilidad	

Tabla 4.1: Ejemplo de tabla de ERS

[27] IEEE STD 830-1998. "Especificaciones de los requisitos del Software", pág. 3.

- Identificador del requisito: RS (Requisito de Software) y dos números.
- Nombre del requisito.
- Descripción del requisito y su objetivo.
- Fuente: Encargado que establece el requisito.
- Prioridad del requisito. Tres posibles valores: alta, media o baja.
- Estabilidad del requisito a lo largo del proyecto. Tres posibles valores: alta, media o baja.
- Prerrequisitos de los que puede llegar a depender el requisito definido.
- Necesidad declarada por la fuente del requisito. Tres posibles valores: alta, media o baja.
- Verificabilidad del requisito. Tres posibles valores: alta, media o baja.

La elección de este formato a la hora de enumerar este tipo de requisitos viene motivada por un objetivo de entendimiento con el cliente y el equipo de desarrollo, público destinatario de la ERS, así que esta estructura y este lenguaje que se utilizarán deberán ser lo más claros y organizados posibles.

Por ello, la organización de la ERS se dividirá en tres tipos de requisitos:

- Requisitos funcionales
- Requisitos de restricción
- Requisitos no funcionales

4.1.- Requisitos de restricción

Este tipo de requisitos corresponden a aquellos que es obligatorio cumplir, puesto que de no ser así, el sistema podría ser inestable.

Identificador	RS-01
Nombre	API REST TMDb
Descripción	La aplicación Android proveerá de una comunicación con la base de datos TMDb mediante una comunicación entre servicios web.
Fuente	Desarrollador
Prioridad	Alta
Estabilidad	Alta
Prerrequisitos	RS-03
Necesidad	Alta
Verificabilidad	Alta

Tabla 4.2: RS-01

Identificador	RS-02
Nombre	REST API Sistema de recomendación
Descripción	La aplicación necesitará de un servicio web que conecte el sistema de recomendación localizado en Azure con la aplicación
Fuente	Desarrollador
Prioridad	Alta
Estabilidad	Alta
Prerrequisitos	RS-03
Necesidad	Alta
Verificabilidad	Alta

Tabla 4.3: RS-02

Identificador	RS-03
Nombre	Conexión móvil a internet
Descripción	Sin conexión móvil a internet, la aplicación no podrá ejecutarse.
Fuente	Desarrollador
Prioridad	Alta
Estabilidad	Media
Prerrequisitos	NA
Necesidad	Alta
Verificabilidad	Alta

Tabla 4.4: RS-03

Identificador	RS-04
Nombre	Permisos para aplicaciones de terceros
Descripción	Permisos de TMDb para el uso de datos del usuario a aplicaciones de terceros, como la que se muestra en este proyecto. De no ser así, el usuario no podrá acceder a las características de la aplicación.
Fuente	Usuario
Prioridad	Alta
Estabilidad	Alta
Prerrequisitos	RS-03
Necesidad	Alta
Verificabilidad	Alta

Tabla 4.5: RS-04

4.2.- Requisitos funcionales

Este tipo de requisitos conformarán el modo en que se debiera comportar el sistema, así como definirá sus componentes o funciones, como pueden ser una

correcta manipulación de los datos o la necesidad de realizar un reporte sobre eventos en la aplicación, entre otros.

Identificador	RS-05
Nombre	Sistema de Recomendación
Descripción	La aplicación ofrecerá al usuario de un sistema de recomendación y este podrá ser utilizado en una vista individual de una película.
Fuente	Desarrollador
Prioridad	Alta
Estabilidad	Media
Prerrequisitos	NA
Necesidad	Alta
Verificabilidad	Media

Tabla 4.6: RS-05

Identificador	RS-06
Nombre	Sistema de Autenticación
Descripción	El usuario deberá de introducir sus datos e identificarse, ya sea como usuario o invitado, en TMDb.
Fuente	Desarrollador
Prioridad	Alta
Estabilidad	Alta
Prerrequisitos	RS-01
Necesidad	Media
Verificabilidad	Alta

Tabla 4.7: RS-06

Identificador	RS-07
Nombre	Galerías de películas favoritas, películas mejor valoradas o listas de películas
Descripción	El usuario obtendrá una galería de películas al elegir cualquiera de las opciones del menú principal.
Fuente	Desarrollador
Prioridad	Media
Estabilidad	Alta
Prerrequisitos	RS-01
Necesidad	Media
Verificabilidad	Alta

Tabla 4.8: RS-07

Identificador	RS-08
Nombre	Acceso a las fichas técnicas de la película
Descripción	En la vista individual de la película, el usuario encontrará los datos de la ficha técnica y artística de la película.
Fuente	Desarrollador
Prioridad	Media
Estabilidad	Media
Prerrequisitos	RS-01
Necesidad	Baja
Verificabilidad	Alta

Tabla 4.9: RS-08

Identificador	RS-09
Nombre	Sistema de almacenamiento
Descripción	La aplicación hace uso de ficheros SQLite que requieren una cantidad de memoria necesaria para almacenar en el móvil.
Fuente	Desarrollador
Prioridad	Media
Estabilidad	Media
Prerrequisitos	NA
Necesidad	Media
Verificabilidad	Media

Tabla 4.10: RS-09

Identificador	RS-10
Nombre	Sistema de búsqueda
Descripción	El usuario tendrá la opción de buscar una película dentro de la base de datos de TMDb.
Fuente	Desarrollador
Prioridad	Media
Estabilidad	Alta
Prerrequisitos	RS-01, RS-07 y RS-08
Necesidad	Media
Verificabilidad	Alta

Tabla 4.11: RS-10

4.3.- Requisitos no funcionales

Por último, estos requisitos corresponden a la definición de atributos de calidad relacionados con propiedades del sistema, como puede ser el almacenamiento en el dispositivo o el método de desarrollo.

Identificador	RS-11
Nombre	Sistema de guardado de películas favoritas en ficheros SQLite
Descripción	Este sistema permitirá reconocer si una película está ya en la lista de favoritas del usuario, indicándolo en pantalla.
Fuente	Desarrollador
Prioridad	Baja
Estabilidad	Alta
Prerrequisitos	RS-09
Necesidad	Baja
Verificabilidad	Alta

Tabla 4.12: RS-11

Identificador	RS-12
Nombre	Algoritmo de correlación
Descripción	A la hora de realizar una recomendación, la aplicación buscará en un fichero SQLite almacenado si existe un identificador de la base de datos de MovieLens que esté relacionado con el identificador de TMDb.
Fuente	Desarrollador
Prioridad	Alta
Estabilidad	Media
Prerrequisitos	RS-02, RS-05 y RS-09
Necesidad	Media
Verificabilidad	Media

Tabla 4.13: RS-12

Identificador	RS-13
Nombre	Desarrollo en Android
Descripción	Durante el desarrollo del proyecto, se utilizará Android Studio como entorno de desarrollo además de Android como plataforma de desarrollo de la aplicación.
Fuente	Desarrollador
Prioridad	Media
Estabilidad	Alta
Prerrequisitos	NA
Necesidad	Baja
Verificabilidad	Alta

Tabla 4.14: RS-13

Capítulo 5: Arquitectura del sistema

En este capítulo, el lector encontrará una descripción al más alto nivel de los sistemas desarrollados en la elaboración del proyecto y las comunicaciones entre estos distintos sistemas.

Como se puede observar en la figura 5.1, se ha decidido estructurar la arquitectura del proyecto en cuatro capas diferenciadas. Así, sería posible la sustitución de algunos de los módulos que componen esta arquitectura por otros, ya sea, por ejemplo, la utilización de un sistema operativo de iOS en lugar del sistema operativo de Android, o la utilización de otro servicio en la nube en vez de Microsoft Azure.

Para una visión más completa del diseño elaborado, a continuación, se muestra la siguiente figura:

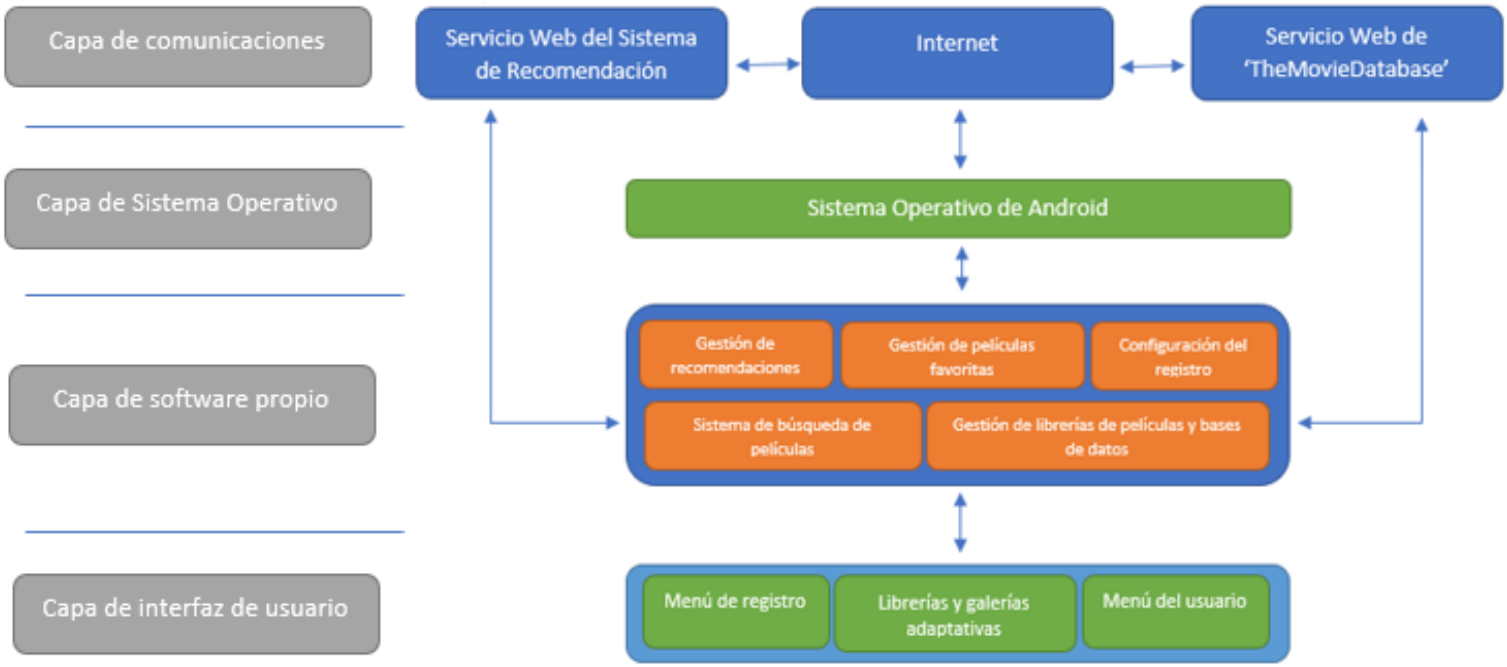

Figura 5.1: Arquitectura del proyecto

5.1.-Capa de interfaz de usuario

Esta capa otorgará al usuario la capacidad de navegar a lo largo de la aplicación Android y de comunicarse con ella, introduciendo datos y recibiéndolos, siendo un medio de entrada y de salida. Respecto a los menús que la conforman, se encuentran:

- Menú de registro. Este menú permitirá al usuario introducir los datos de registro de la red social de 'TMDb', que a su vez otorgará acceso a las diferentes librerías de películas que se encuentran en su base de datos.
- Librerías y galerías adaptativas. Entre las opciones que proporciona la aplicación se encuentran la opción de ver las películas que el usuario registrado como favoritas o incluso las películas mejor valoradas en 'TMDb'. Puesto que el número de películas que se encuentran en este tipo de listas puede variar, se ha apostado por un diseño dinámico que se ajusta a la cantidad de estas. En este tipo de galerías, será posible escoger entre las diversas películas y entrar a un menú individual en el que se ofrecen más detalles y características del ítem en cuestión.
- Menú de usuario. Permite una navegación cómoda, flexible e intuitiva entre las diversas opciones que da la aplicación Android, como, por ejemplo, la búsqueda de películas por título o identificador.

5.2.- Capa de software propio

En esta capa se agrupan todos los módulos de software desarrollados durante la implementación de la aplicación Android. Estos, a su vez, hacen uso de las diversas funcionalidades que aporta el sistema operativo de Android, entre las que destaca el acceso a Internet. Respecto a las tareas que se han implementado, a continuación, se muestran algunas de ellas:

- Gestión de recomendaciones. Este apartado permitirá en el menú individual de cada película, la opción de marcarla como 'película favorita'. Esta opción enviará los datos pertinentes mediante servicio web a 'TMDb' y se registrará.
- Gestión de películas favoritas. A continuación de haberla marcado como favorita, si el usuario navegara al menú de películas recomendadas, podría visualizar allí todas las películas registradas como favoritas en 'TMDb'.
- Configuración del registro. Esta tarea permitirá introducir los datos de inicio de sesión en la aplicación, los cuales serán enviados a la base de datos y, mediante un intercambio de 'tokens', se oficializará el inicio de sesión en la aplicación, lo que permitirá el intercambio de datos posterior con el servicio web directamente desde la cuenta del usuario de 'TMDb'.
- Sistema de búsqueda de películas. En este módulo se ha configurado una opción que permitirá al usuario la búsqueda de un ítem mediante una consulta de texto, que se enviará a la base de datos.
- Gestión de películas y bases de datos. Durante todo este proceso, se hará uso de las librerías de SQLite que Android proporciona mediante su sistema operativo. Así, se registrarán en bases de datos nativas ciertas librerías de películas. Por ejemplo, siempre que el usuario navegue por el menú de películas favoritas, se hará una copia de esta galería a una base de datos SQLite. Otro de los ejemplos que se pueden encontrar en esta aplicación es el uso de una base de datos SQLite para comparar los distintos identificadores de las películas. Al haber configurado el sistema de recomendación con una base de datos de películas propia, se hará uso de esta base de datos para comparar los identificadores que vengan del servicio web del sistema de recomendación, con los identificadores de las películas en la base de datos de 'TMDb'.

5.3.- Capa de Sistema Operativo

Las funcionalidades de esta capa son las propias del sistema operativo de Android. Durante el capítulo 'Estado del arte' se han detallado muchas de las características de este sistema operativo.

5.4.- Capa de comunicaciones

En esta capa se encuentran con varios de los módulos que proporcionan el acceso online a la base de datos de películas. Además de Internet, se encuentran:

- Servicio web del sistema de recomendación. Como bien se ha comentado antes, después de haber estructurado y formado el sistema de recomendación en el servicio en la nube Microsoft Azure, se ha hecho uso de un servicio web para el intercambio de datos entre ambos sistemas, el de la aplicación y el de Azure. Así, en los menús individualizados de cada película, será posible navegar hasta las recomendaciones de este ítem y acceder a un menú diferente que muestre la galería de películas que el sistema de recomendación devuelve.
- Servicio web de 'TheMovieDatabase'. Esta base de datos será, posiblemente, una pieza central en este proyecto. Como piedra angular, nos dará acceso a la infinidad de películas que existen registradas en una de las mayores bases de datos del mundo.

Capítulo 6: Diseño

En este capítulo, se hará un desglose con el diseño de los tres elementos más importantes de la arquitectura del proyecto antes mencionada: el sistema de recomendación en Azure, la aplicación móvil en Android y el servicio web de TMDb. Sin embargo, el punto inicial de este capítulo será describir la cantidad masiva de datos que se han usado a la hora de montar el sistema de recomendación.

6.1.- Datasets de MovieLens

Para el proyecto, se han estudiado varias gamas de conjuntos de datos que encajaran con el objetivo inicial de realizar el sistema de recomendación de películas. Así, se encuentra GroupLens como referente en este tipo de proyectos. GroupLens es un equipo de investigación del departamento de Ingeniería de la Computación en la Universidad de Minnesota. Como equipo, han logrado grandes avances en diversos campos como motores de recomendación, tecnologías móviles o comunicaciones. Así, tienen un proyecto abierto, cuyo nombre es

MovieLens, en el que comparten de manera gratuita unos datasets[28] para el estudio de motores de recomendación.

Respecto a los datasets, las cifras que se manejan alcanzan 20 millones de valoraciones, con 465.000 etiquetaciones en más de 27.000 películas por 138.000 usuarios, datando la última actualización en octubre de 2016. Durante este proyecto, de entre todos los datasets de los que disponen, se ha hecho uso principalmente de tres: ratings.csv, movies.csv y links.csv.

6.1.1- Estructura de datos de movies.csv

Este fichero contiene la información primordial de la película. Cada línea del fichero corresponde a una película distinta y corresponde al siguiente formato:

$$movieId(year), title, genres$$

Los géneros (pues cada película puede tener más de uno asignado) son dispuestos con una separación de tubería. Entre ellos, se encontrarán:

- Acción
- Aventura
- Animación
- Infantil
- Comedia
- Crimen
- Documental
- Drama
- Fantasía
- Cine Negro
- Terror
- Musical
- Misterio
- Romance
- Ciencia-ficción
- Thriller
- Guerra
- Western

6.1.2.- Estructura de datos de ratings.csv

En este fichero se encuentran todas las valoraciones de las distintas películas. Cada línea del fichero está compuesta por una valoración y dispuesta de la siguiente manera:

[28] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872

$$userId, movieId, rating, timestamp$$

Los usuarios de MovieLens han sido incluidos de manera aleatoria, así como sus identificadores. Así, el fichero ha sido ordenado por identificador de usuario y posteriormente por el identificador de película.

Las valoraciones siguen una escala de 5 estrellas, con granularidad de media estrella (de 0,5 a 5 estrellas). La marca de tiempo está representada en segundos desde el 1 de enero de 1970 según UTC.

6.1.3.- Estructura de datos de links.csv

En este fichero se encuentran las correlaciones entre los diversos identificadores que se pueden utilizar para gestionar las películas:

$$movieId, imdbId, tmdbId$$

El primer identificador es el identificador que se ha utilizado en https://movielens.org. El segundo, es el identificador que se puede encontrar en la base de datos de IMDb, , https://www.imdb.com. Por último, tmdbId indica el identificador en la base de datos que más se ha utilizado a lo largo del proyecto, TMDb, https://www.themoviedb.org.

6.2.- Sistema de recomendación en Azure

6.2.1.- Arquitectura en Azure

El sistema de recomendación diseñado en la nube se compone a su vez de dos esquemas distintos. El primero es el modelo de entrenamiento. En él, se diseña el sistema a entrenar con los datos de MovieLens. Se muestra a continuación.

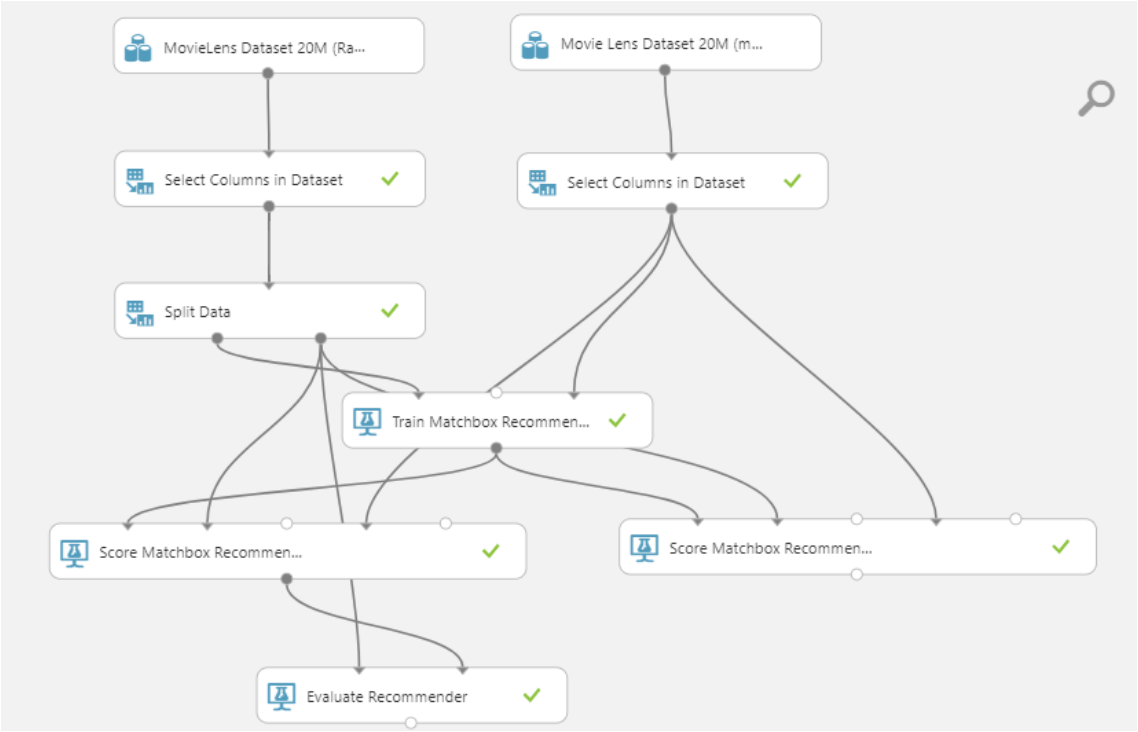

Figura 6.1: Diseño del modelo de entrenamiento del sistema de recomendación

En la parte superior de la figura se puede observar los dos conjuntos de datos que se han escogido para entrenar el sistema. En la izquierda, el fichero de ratings.csv que recoge las valoraciones de las distintas películas que se incluyen en estos conjuntos de datos. A la derecha, el listado de películas que aparece en movies.csv.

Ambos conjuntos de datos serán procesados para el correcto entrenamiento del sistema, comenzando por elegir las columnas que se utilizarán en nuestro diseño. Para ambos conjuntos de datos:

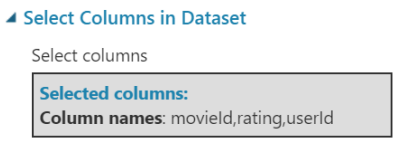

Figura 6.2: Selección de columnas en ratings.csv

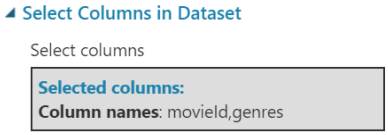

Figura 6.3: Selección de columnas en movies.csv

Al realizar la selección de columnas necesarias para ambos conjuntos de datos, el siguiente paso para el conjunto de datos de las valoraciones será dividir la información en dos segmentos: uno destinado al entrenamiento de datos y el

segundo que permitirá la evaluación del modelo. Así, el módulo que indica "Split data", tendrá las siguientes características:

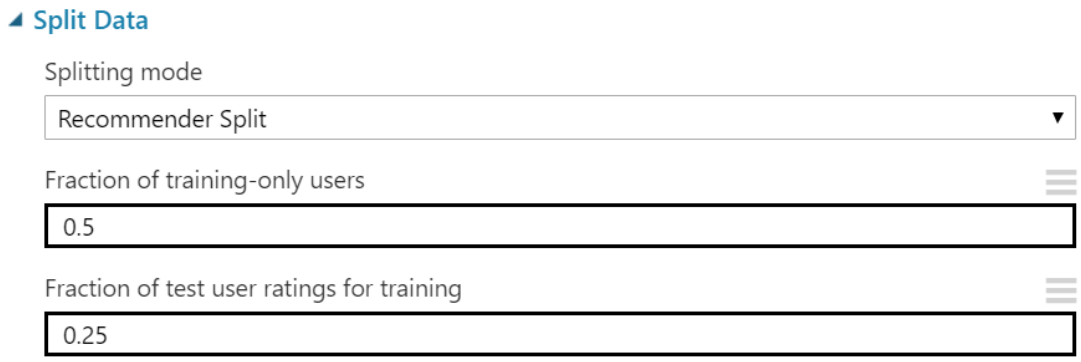

Figura 6.4: División de datos de las columnas seleccionadas de ratings.csv

El modo de división elegido ha sido "Recommender Split", asignando el 50% de los datos para un posible uso de entrenamiento y no para una prueba de la valoración del modelo. El 25% que indica la anterior figura, ilustrarán la porción de valoraciones de usuarios recogidos que pueden ser utilizados para entrenamiento. Así, la salida número 1 que se observa en la figura 6.1, mantendrá esas filas seleccionadas, mientras que la salida número 2 formarán el conjunto restante.

Esta salida número 1, al igual que la salida de la selección de columnas del conjunto de datos de movies.csv, llegarán como entradas del módulo "Train Matchbox Recommender". Este módulo, especializado en sistemas de recomendación, lee conjuntos de datos relacionados con usuario-ítem-valoración y devuelve un modelo de sistema de recomendación. Así, al haber introducido un enfoque de contenido (puesto que los datos recogidos de movies.csv incluyen también el género) y un enfoque colaborativo (con los usuarios de la comunidad y sus valoraciones de los ítems), este módulo combina ambos enfoques para lograr un sistema de recomendación híbrido, mediante un sistema de recomendación Bayesiano, cuya implementación se detallará en el siguiente capítulo.

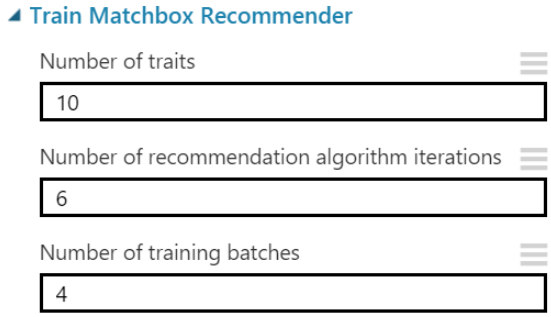

Figura 6.5: Módulo de Train Matchbox Recommender

La salida de este módulo corresponderá al sistema de recomendación que se duplicará para realizar dos evaluaciones en los módulos que se indicaba en la figura inicial, los correspondientes al "Score Matchbox Recommender".

Al módulo de la izquierda servirán de entradas: el conjunto de datos de las valoraciones junto al sistema de recomendación y el conjunto de datos referido a los ítems, realizando una predicción de valoración o "Rating Prediction", en el que el modelo predecirá cómo un usuario de la comunidad valorará cierto ítem. Azure permite realizar una observación de estas valoraciones, como se ilustra en la siguiente figura.

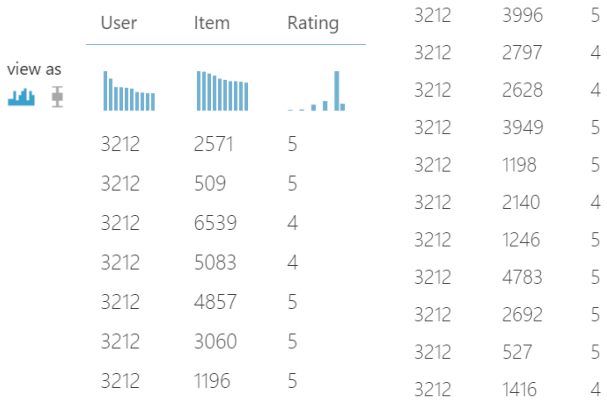

Figura 6.6: Ejemplo de predicciones de usuarios a ítems

En el módulo de valoración de la derecha, se ha realizado una valoración en cuanto a una posible recomendación de ítems. El modelo utilizará la información proporcionada de los ítems y de los usuarios para generar una lista de ítems con cierta atracción para cada usuario. Así, la valoración del modelo se resume en la siguiente figura:

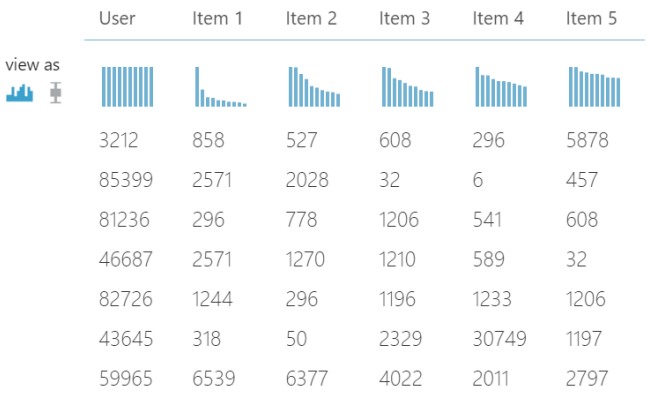

Figura 6.7: Visualización de las recomendaciones a cada usuario

En la figura 6.7 se puede observar como el modelo recomendará, por ejemplo, al usuario 3212 los cinco ítems más afines a sus gustos, siendo el más recomendable el Ítem 1 y el menos recomendable, Ítem 5.

Por último, se encuentra el módulo referido a "Evaluate Recommender", que medirá la precisión de las predicciones realizadas por el modelo. A la hora de utilizar este módulo, será posible evaluar tanto la predicción de valoraciones de ítems como la precisión de las recomendaciones a cada usuario. Las variables necesarias que se introducen como entrada a este módulo serán: el conjunto de datos de prueba, en forma de usuario-ítem-valoración, y el conjunto de datos

evaluados, que contienen las predicciones generadas por el modelo de recomendación. Así, este módulo calculará el error medio absoluto (aquí indicado como MAE) y la raíz cuadrada del error cuadrático medio (indicado aquí en inglés como RMSE). Así, cuando se evalúa la predicción de valoraciones, se encuentran los siguientes resultados:

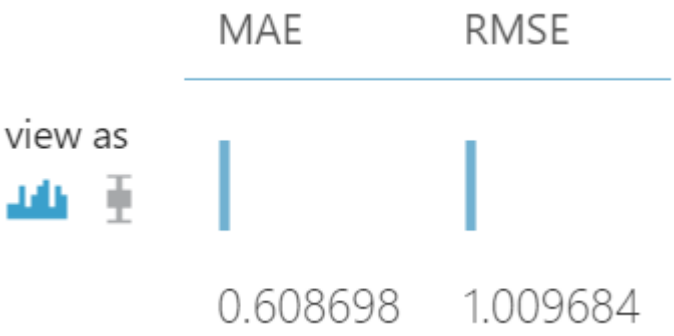

Figura 6.8: Métrica de la evaluación del modelo

En comparación, el valor de RMSE para el concurso llamado BellKor's Pragmatic Chaos, creado por Netflix y que describe en el capítulo "Estado del Arte", fue medido en 0.8533.

En el siguiente capítulo, se detallará la implementación en Azure de este sistema de recomendación y la creación del servicio web por el que se comunicará con la aplicación Android.

6.3.- Diseño de interacción

En este apartado se comentará la disposición y el diseño de la navegación del usuario a lo largo de la aplicación.

En primer lugar, al iniciar la aplicación, el usuario se encuentra ante esta vista:

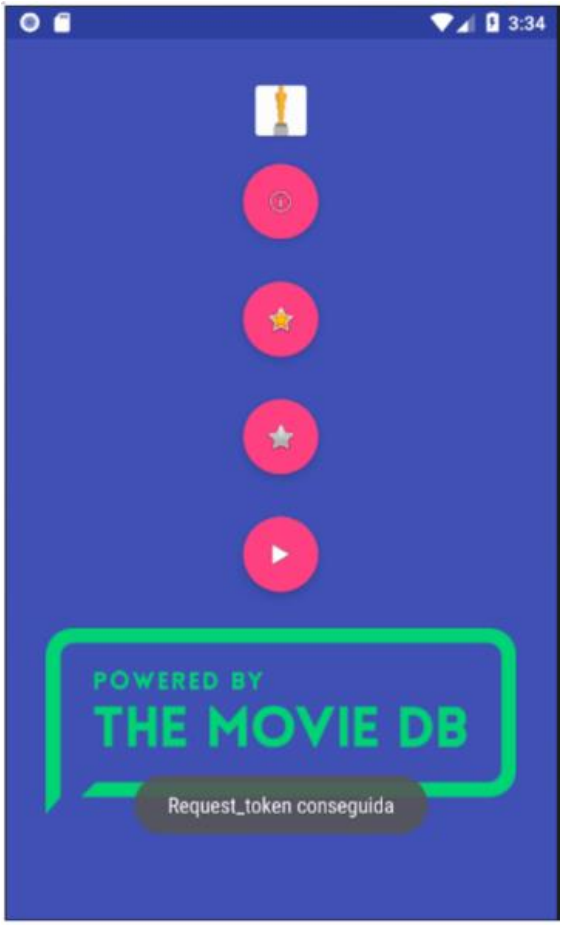

Figura 6.9: Vista de autenticación

En ella, el usuario tendrá la posibilidad de autenticarse como usuario de la base de datos TheMovieDatabase. Como se puede observar, en primer lugar la aplicación dispondrá automáticamente de una clave para realizar la autenticación. La aplicación permitirá autenticarse como usuario normal (los dos primeros botones) o usuario invitado (el tercer botón). Presionando sobre el primer botón, la aplicación redirige al usuario hacia una vista Web, llamada así porque equivale a un navegador web.

En esa vista, el usuario podrá ver lo siguiente:

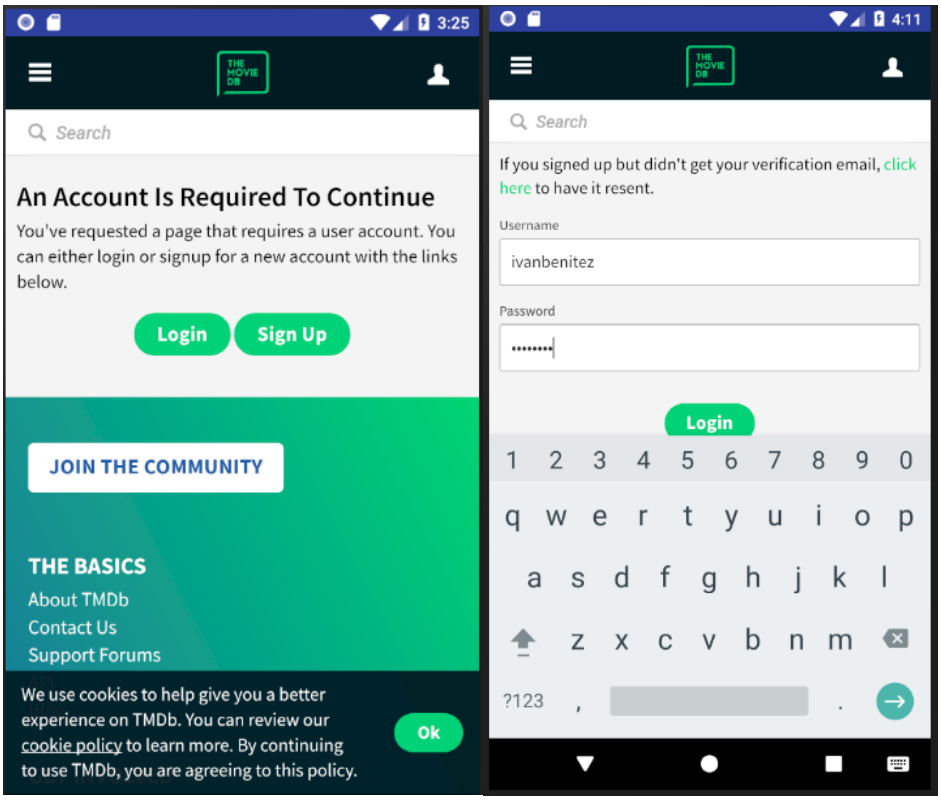

Figura 6.10: Vistas de la autenticación en el navegador incrustado

Presionando en el botón de Login e introduciendo los datos pertinentes, el navegador web, desde la página de TheMovieDatabase, preguntará al usuario si quiere confirmar la autenticación para una aplicación externa como es la que se presenta:

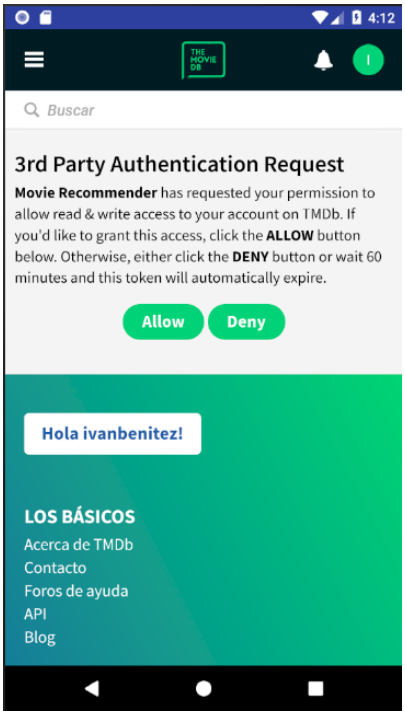

Figura 6.11: Vista del requerimiento de autenticación para la app

Al presionar en el botón en el que aparece “Allow”, el usuario vuelve a la vista de autenticación (Figura 6.10), y deberá pulsar el segundo botón para conseguir así una clave de sesión correcta. Entonces, si el usuario presiona el último botón de los que se mostraban antes, llegará a la vista del menú principal.

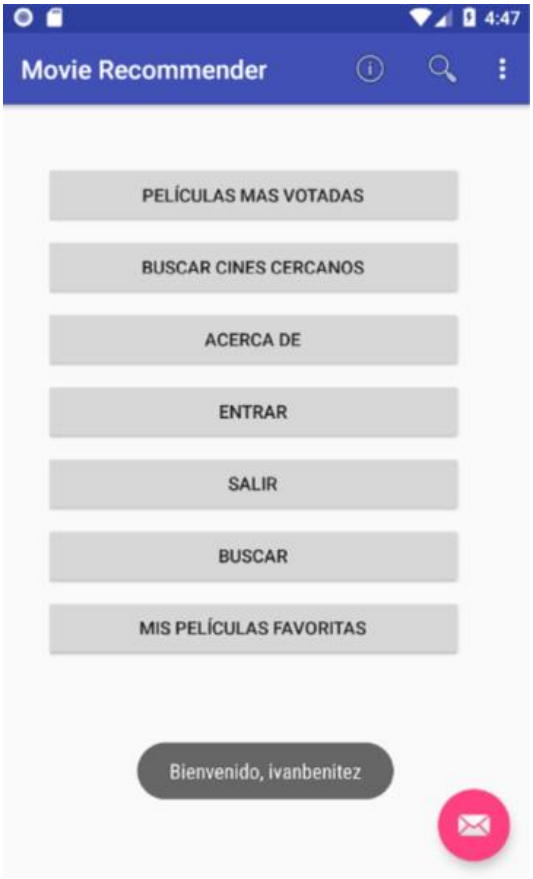

Figura 6.12: Vista del menú principal

En este menú, como bien aparece reflejado en los distintos botones, se dispondrán las distintas opciones que el usuario tiene en la aplicación. Si bien el diseño puede parecer simple, esto se presenta también como futura mejora para un diseño más atractivo para los usuarios.

Al presionar en el botón “Películas más votadas”, el usuario entrará en una vista en la que se presenta una galería de las películas mejor valoradas por los usuarios de TheMovieDatabase.

La disposición de las películas que conforman las galerías ha sido posible gracias a una librería llamada RecyclerView, que permite replicar vistas singulares tantas veces como elementos se tengan en un vector de objetos. Un ejemplo de ello se puede encontrar al presionar el botón “Mis películas favoritas”, el cual dirigirá al usuario a la siguiente vista:

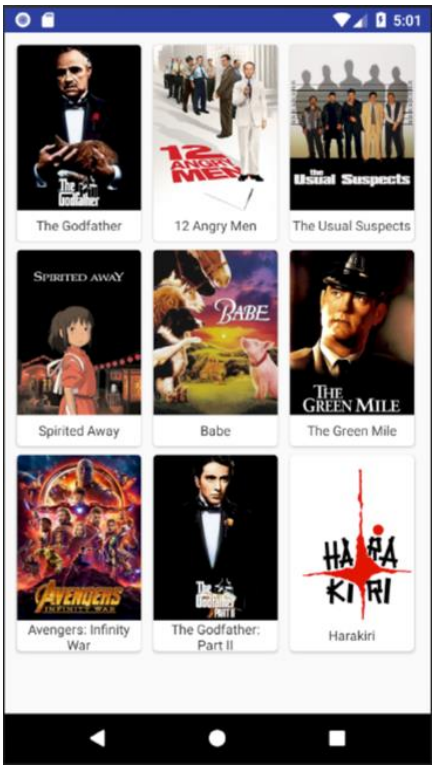

Figura 6.13: Vista de “Mis Películas Favoritas”

En esta sección, se pueden encontrar las películas que el usuario de “TMDb” ha seleccionado como favoritas dentro de la base de datos y que también recoge la aplicación. Al presionar una de las películas dispuestas, la aplicación tendrá esta forma:

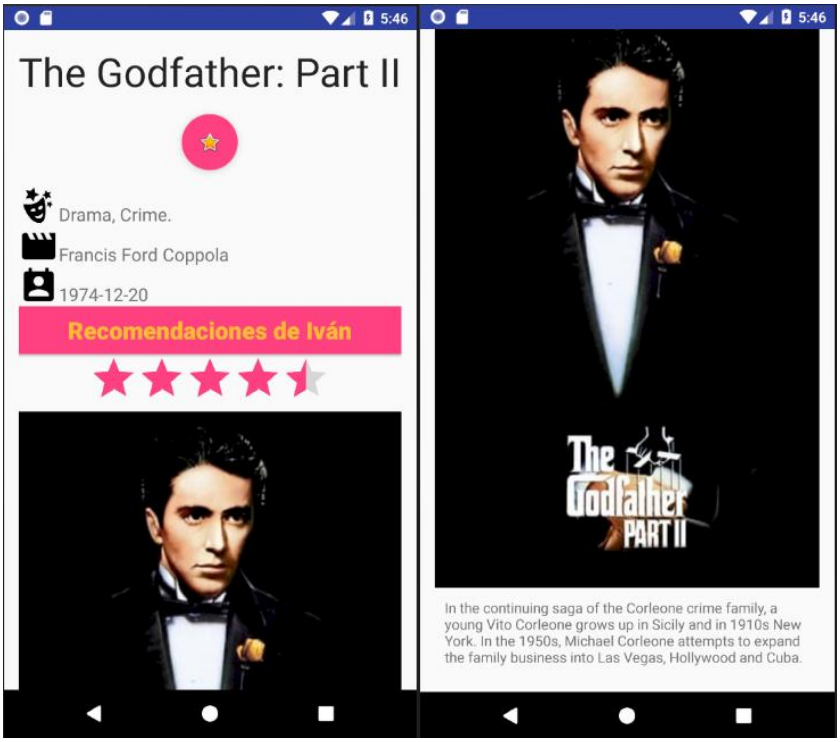

Figura 6.14: Vistas de una película individual

En esta vista, se pueden observar diversos detalles que se mostrarán de la película, como el título, los géneros, el director, la fecha de estreno, una valoración en forma de estrellas y la descripción del argumento de la película.

Cabe resaltar que, bajo el título de la película, se puede observar un botón con una estrella en su interior. Este botón permite reconocer si la película forma parte de la galería de películas favoritas del usuario o no. Por tanto, al pulsar en ese botón, será posible introducirla en esa categoría, en el caso en que la estrella se disponga de color amarillo, o retirarla, en el caso en que la estrella no tenga color.

Además, la integración del sistema de recomendación de Azure se ha dispuesto de la siguiente manera. Al presionar el botón "Recomendaciones de Iván", la aplicación realizará la llamada al servicio web montado en Azure y devolverá así los datos necesarios para disponer una galería en el que se incluyan las seis películas que el sistema de recomendación dispone para esta película. Con este ejemplo, al presionar en ese botón aparecerá lo siguiente:

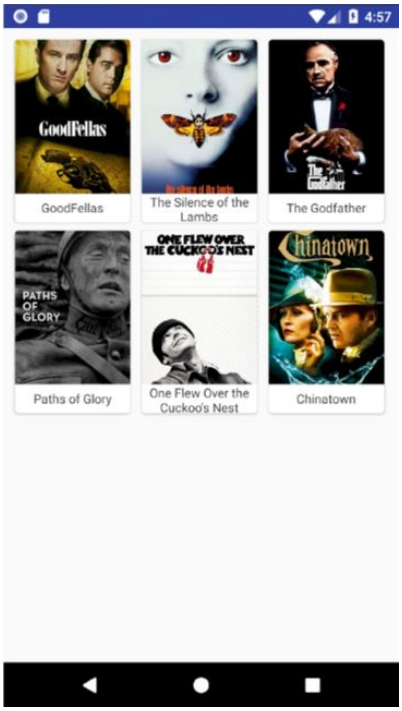

Figura 6.15: Vista de las recomendaciones de Azure

Cabe destacar el siguiente aspecto: pese a que la base de datos de películas que proporciona MovieLens es muy amplia, es muy posible que algunas películas no se encuentren en ella, por ejemplo, las más modernas. ¿Cómo es posible entonces enviar una un ítem al sistema de recomendación de Azure si ese no existe?

En caso de no existir ese ítem dentro de la base de datos de MovieLens, en vez de mostrar en pantalla las recomendaciones que proporcionan MovieLens, las películas que aparecerán en pantalla seguirán a las películas recomendadas por el sistema de TMDb mediante una llamada al servicio web. Así, por ejemplo, para

una película moderna como es “Avengers: Infinity War”, las recomendaciones serán las siguientes:

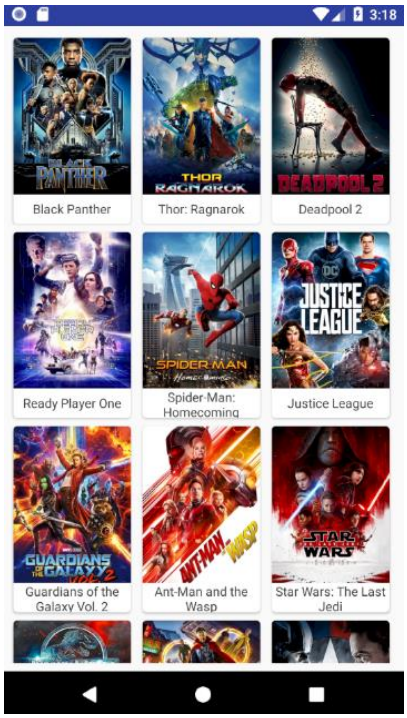

Figura 6.16: Vista de las recomendaciones de TMDb

Se puede observar que el número de ítems recomendados es mayor y esto se debe a la limitación puesta en el sistema de recomendación de Azure de dar salida únicamente a 6 posibles ítems recomendados.

Volviendo al menú principal, otra de las opciones que la aplicación permite realizar a el usuario es la de buscar una película en la base de datos. Al presionar el botón “Buscar”, la aplicación tiene esta vista:

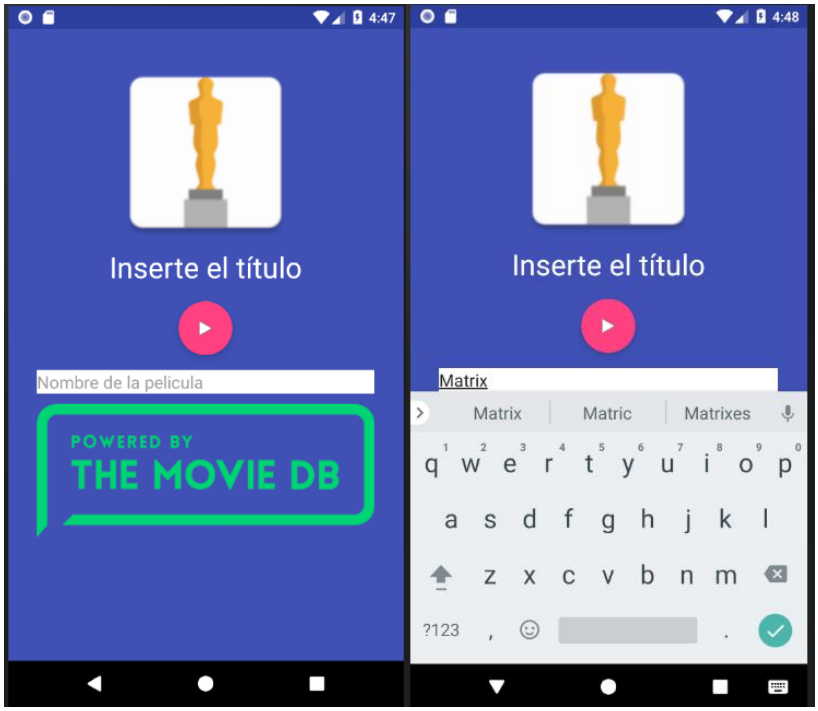

Figura 6.17: Vista de la búsqueda de películas

La aplicación permitirá introducir el título de una película y buscará todos los resultados cercanos en la base de datos “TheMovieDatabase”. Posteriormente, al presionar en el botón de color rosa, la aplicación redirigirá al usuario a una galería con todos los resultados encontrados:

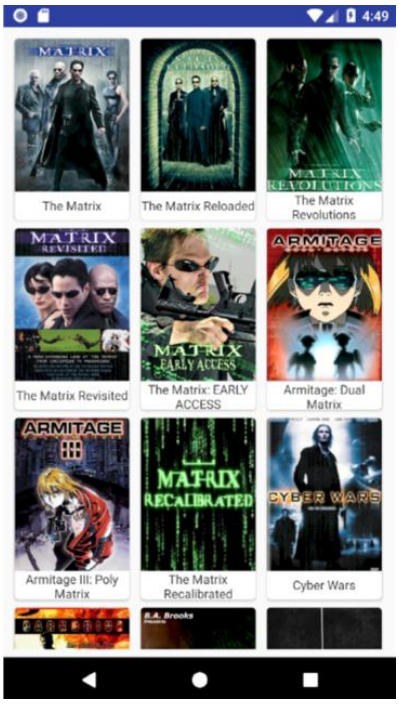

Figura 6.18: Vista de la galería de resultados de la búsqueda

Y al presionar en cualquiera de los ítems que aparecen en la galería, se volverá a una vista individual de la película (similar a la figura 6.15):

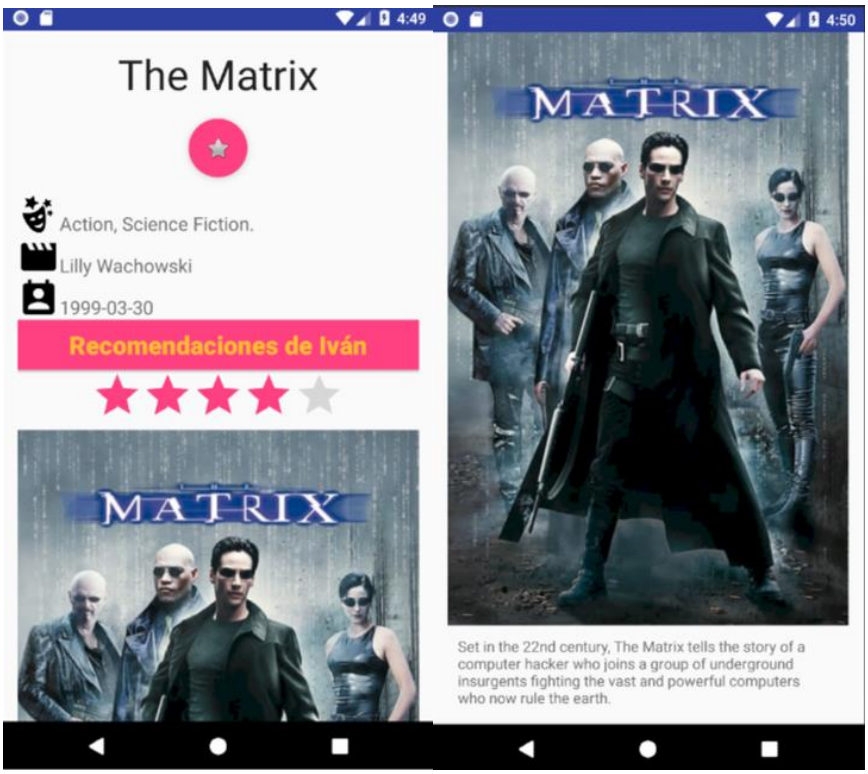

Figura 6.19: Vista de la película buscada

6.4.- Módulos de la aplicación

En esta sección se detallarán los módulos que componen la aplicación en su totalidad y se indicarán las clases principales que componen cada módulo, sin entrar en la implementación del código.

Para empezar, se presenta a continuación un diagrama que muestra la arquitectura de la aplicación por módulos.

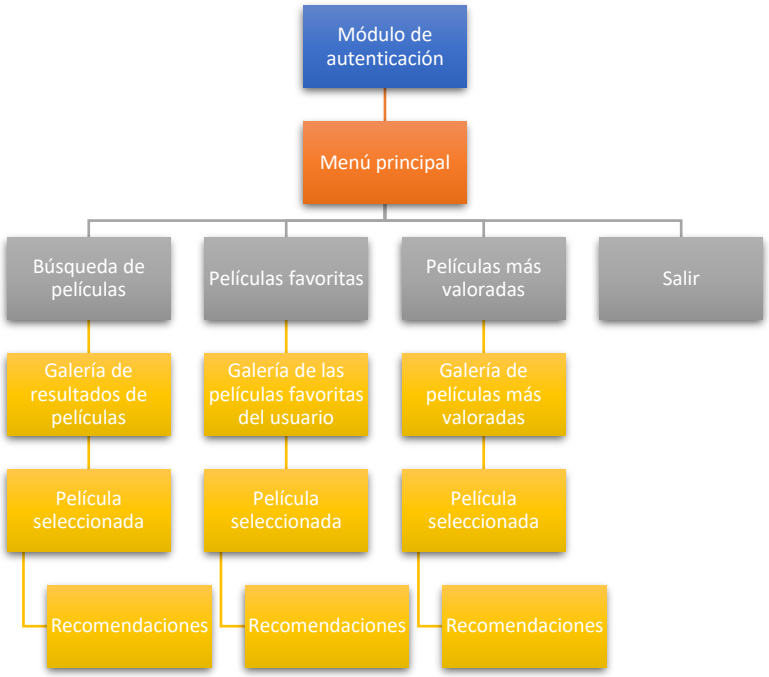

Figura 6.20: Diagrama de la arquitectura de la aplicación

6.4.1.- Módulo de autenticación

En este módulo, el usuario se autenticará con sus credenciales en la base de datos de TMDb, aunque la aplicación ofrece también una segunda opción para entrar como usuario invitado. Para la primera opción, "TMDb" sigue el siguiente proceso:

1. El usuario recibe una clave inicial que le dará permiso para solicitar su clave de sesión.
2. El usuario deberá permitir el acceso a la base de datos por parte de la aplicación. Para ello, deberá introducir sus claves de ingreso.
3. Al permitirlo, el usuario recibirá la clave de identificación de sesión que le permitirá tener acceso libre a todas las características que ofrece "TMDb".

La clase inicial donde comienza la aplicación es Logon.java. La clase ha sido configurada para que, al iniciarse, pida, mediante una llamada al servicio web, la clave inicial (también denominada "request token").

Se ha desarrollado a su vez una clase paralela que se denomina, WebActivity.class, en la que se hará uso de WebView, una aplicación que hace de visor de páginas web y que es nativo de Android. Esto permitirá cargar una URL en la que, después de autenticarse con su nombre de usuario y contraseña, el

usuario podrá permitir el acceso a la base de datos como se indica en el paso 2. Al permitir el acceso, el usuario volverá a la página de autenticación.

Otra vez de vuelta, el usuario deberá presionar el segundo botón para solicitar la clave de identificación de sesión. Así, el usuario ya podrá dirigirse al menú principal de la aplicación.

Cabe destacar la importancia que tiene la clave de identificación de sesión, pues en muchas de las llamadas a la API que se harán a continuación, es solicitada para identificar al usuario en nombre del cual se realizan las llamadas.

Para la opción en la que el usuario se autentica como invitado, el proceso difiere un poco:

1. El usuario recibe también una clave inicial que le dará permiso para solicitar su clave de sesión.
2. Al no ser usuario de la base de datos TMDb, no es necesario dar acceso a la aplicación, puesto que el usuario invitado no tendrá las mismas características que un usuario normal.
3. El usuario entonces podrá recibir una clave de identificación de sesión de invitado, que limitará las características de la aplicación.

La diferencia entre un usuario normal y un usuario invitado radica en la posibilidad de mostrar las películas favoritas del primero, ya que el segundo no dispone de perfil para guardarlas. Así, tampoco le será posible marcar películas como favoritas para que se integren en esa lista.

6.4.2.- Menú principal

La clase que presenta las distintas opciones de uso de la aplicación se denomina MainActivity.class. Uno de los principales procesos que comenzará al llegar a este módulo, será la de una llamada a la API con la que se conseguirán los datos de sesión del usuario, con lo que la aplicación logrará identificarle por su nombre de usuario o como “Invitado”, limitándole las características antes mencionadas.

Esta clase sirve de portal para el inicio de las diferentes actividades que el usuario elegirá al presionar un botón u otro.

6.4.3.- Búsqueda de películas

Este módulo es el único que difiere respecto a los módulos de Películas Recomendadas o Mis películas favoritas. La clase que gestiona la parte inicial de este módulo se denomina BuscarPelicula.class. En ella, se recogerá una línea de texto o “string”, que el usuario introducirá y se pasará como argumento a la

siguiente actividad, en la que mediante una llamada a la API, se recogerá una lista de películas que responderán como elementos relacionados con ese texto introducido. La galería de películas que se presentará se describe en la siguiente sección.

6.4.4.- Galerías de Películas (Películas más votadas, Películas favoritas o Resultado de la búsqueda)

En este módulo, se presentan al usuario las galerías de películas que haya requerido, sean el resultado de la búsqueda que antes se mencionaba, la lista de películas favoritas del usuario en la base de datos "TMDb" o las películas con mejor valoración en esa misma base de datos.

Pese a que las clases principales de los tres módulos son distintas (ListaBusqueda.class, MyFavMovies.class y VistaPeliculaActivity.class, respectivamente), todas tienen una funcionalidad similar. Primero, realizan una llamada al servicio web correspondiente, recogiendo la galería de películas que se requieran. Al obtenerla, se hace uso de una clase común que tiene por nombre MoviesAdapter.class. Gracias a la librería de RecyclerView que se menciona en el apartado anterior de navegación, se ha hecho uso de un diseño único de ítem y se ha conseguido replicar tantas veces como elementos en la galería haya. Para ello, se hace uso del adaptador MoviesAdapter.class, que dispone los elementos recogidos por la API y los presenta al usuario como elementos CardView, otro widget que Android proporciona para mostrar la información en un formato de carta.

6.4.5.- Película Individual

Al presionar uno de los elementos dispuestos como cartas dentro de la galería, el usuario accederá a una vista individual del ítem. En esta vista individual se hace uso también de un adaptador, MoviesAdapterInfo.class, que recoge los datos de la película seleccionada y mediante distintas llamadas a la API, recogerá el título de la película, su fecha de estreno, su póster, los géneros a los que pertenece, el director de la película y una descripción del argumento de la película.

En este adaptador, es posible también mediante realizar una llamada al servicio web y enviar el identificador de la película, marcándola como favorita e integrándola en la galería de películas favoritas anteriormente mencionado.

Por último, cabe destacar que a esta clase se le añade la dificultad de una llamada a la clase MyRecommendedMovies.class, y que se realiza cuando se requieren películas recomendadas que le podrían gustar al usuario al haber seleccionado este ítem. En el hilo de procesos que realiza esta clase, se identifica a la película mediante su identificador y se realiza una búsqueda mediante SQLite en una pequeña base de datos que existe integrada en la aplicación, denominada linkslite.sqlite, que no deja de ser el archivo links.csv que se mencionaba al inicio del capítulo y que se ha convertido a SQLite. Mediante uso de lenguaje de base de datos, se buscará una relación entre el identificador que se recoge de "TMDb"

y el identificador que poseen los distintos ítems de la base de datos de MovieLens que se han mencionado al inicio del capítulo.

Siguiendo el flujo del proceso, en el caso en que esa película no se encuentre en la base de datos de MovieLens, se realizará una llamada al servicio web con el objetivo de recibir una lista de películas que el sistema de "TMDb" recomienda según la películas seleccionada y el usuario volvería a una galería de películas como la que se mencionaban en la anterior sección.

En el caso en que el identificador sí se encuentre en la base de datos, se recogerán los resultados del sistema de recomendación que se ha provisto en Azure, también mediante llamadas a un servicio web. La dificultad que reside en el hecho de que esas llamadas a la API son asíncronas, se dispondrán de forma secuencial para la obtención de las 6 películas que Azure entrega a la aplicación. Con esos resultados, el usuario vuelve a hacer uso de las galerías de películas con MoviesAdapter.class. y RecyclerView, cerrando el círculo.

6.5.- Configuración final de las clases

Finalmente, a continuación se muestra un diseño de la arquitectura de clases a la que se ha ajustado la aplicación Android. Pese a que se muestra de una forma más simplificada, en el siguiente capítulo se mostrarán algunas de las clases auxiliares que se han utilizado para complementar esta estructura (la interfaz donde se enumeran las llamadas a las APIs, las clases genéricas que responden a objetos recuperados de las APIs, etc.).

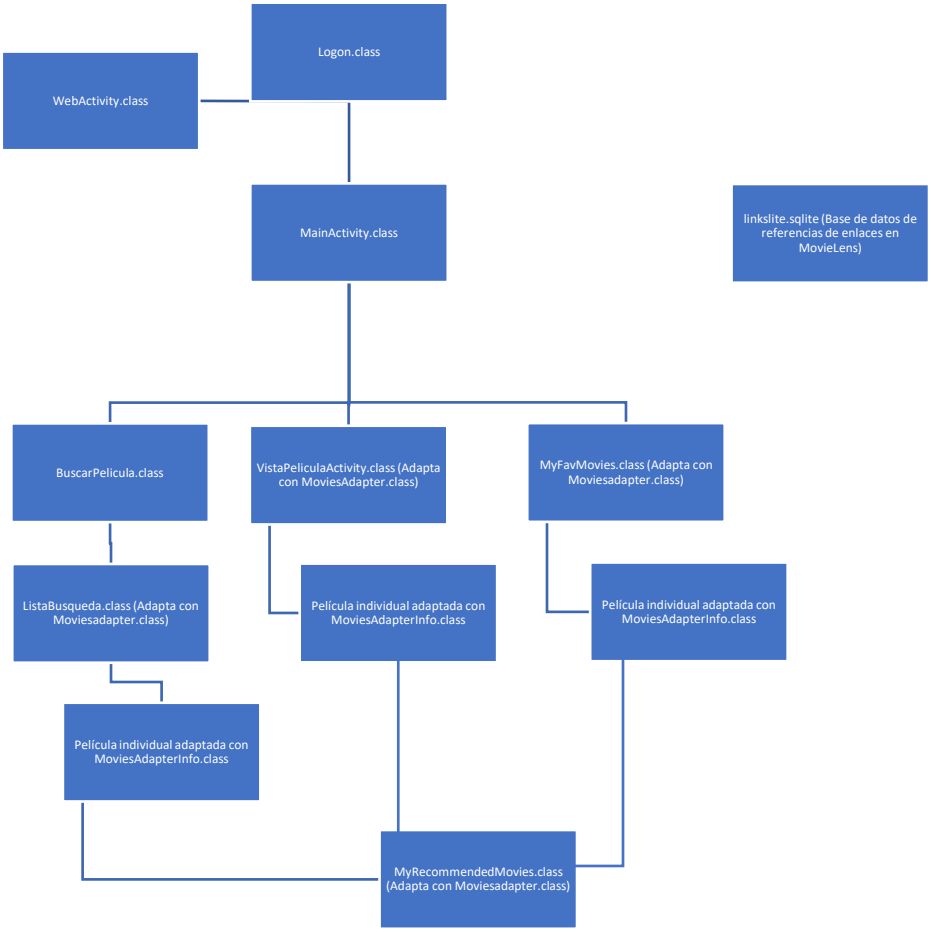

Figura 6.21: Diagrama de la arquitectura de clases de la aplicación

Capítulo 7: Implementación

A lo largo de este capítulo, se mencionarán los aspectos más técnicos relacionados con cómo se ha ajustado el diseño al proyecto final. Se discuten entre otros aspectos: la programación de la conexión entre el servidor web de "TMDb" y la aplicación, y la implementación del sistema de recomendación de la nube dentro de la aplicación.

7.1.- Implementación del servicio web de "TMDb"

Antes de enumerar las distintas llamadas que la aplicación hará a los servicios web, se ha de mencionar que se usará "Retrofit". "Retrofit" es un cliente REST desarrollado por Square para Android y Java, convirtiendo así una API HTTP en una interfaz Java. Añadiendo las dependencias oportunas al gradle del proyecto, este cliente REST permitirá realizar peticiones de cualquier tipo, obteniendo una respuesta que devolverá como un Plain Old Java Object (POJO).

Para ello, uno de los primeros pasos que se ha realizado ha sido incluir todas las llamadas a las diversas APIs en una interfaz denominada MovieApiService.java. A continuación se enumeran las llamadas, las cuales contendrán de forma común la clave de la API que otorga "TMDb" para el uso de la misma.

7.1.1.- Request Token

```
@GET("authentication/token/new")
Call<RequestToken> getNewToken(
@Query("api_key") String api_key);
```

Se trata de la primera llamada que la aplicación realiza a la API. En ella se requiere una clave para solicitar la autenticación del usuario al iniciar la aplicación. Así, se ha creado una clase con el nombre de RequestToken, que recoge los campos necesarios: la clave y cuándo expira.

7.1.2.- Identificador de sesión

```
@GET("authentication/session/new")
Call<RequestSessionID> getSessionID(
@Query("api_key") String api_key,
@Query("request_token")String request_token);
```

Para esta llamada GET, ya con la clave anterior y habiendo accedido y dado permiso a la base de datos para poder utilizar una aplicación de terceros como esta, se recoge un objeto llamado RequestSessionID que proporcionará un identificador de sesión que se utilizará para algunas características de la aplicación.

7.1.3.- Películas más valoradas

```
@GET("movie/top_rated")
Call<MovieResponse> getTopRatedMovies(
@Query("api_key") String apiKey);
```

Esta llamada GET recibe así un objeto convertido a POJO que se ha creado y denominado MovieResponse.java. Esta clase contiene internamente un vector donde se irán introduciendo las distintas películas que recogen las listas requeridas.

7.1.4.- Buscar Películas

```
@GET("search/movie")
Call<MovieResponse> getMovies(
@Query("api_key") String apiKey,
@Query("query") String query);
```

Para esta llamada se utilizará un objeto String que recoge la línea de texto que el usuario escribe cuando desea buscar una película. También se recoge en un objeto MovieResponse.

7.1.5.- Mis Películas favoritas

```
@GET("account/{account_id}/favorite/movies")
Call<MovieResponse> getFavMovies(
@Path("account_id") int account_id,
@Query("api_key") String api_key,
@Query("session_id") String session_id);
```

Se recoge también un objeto MovieResponse en el cual se introducen todas las películas que el usuario ha marcado como favoritas en la base de datos “TMDb”, para ello es necesario el identificador de sesión que antes se mencionaba y que aquí se pasa como argumento a la vez que el identificador de la cuenta del usuario.

7.1.6.- Marcar película como favorita

```
@POST("account/{account_id}/favorite")
Call<SuccessResponse> postFavMovie(
@Path("account_id") int account_id,
@Query("api_key") String api_key,
@Query("session_id") String session_id,
@Header("content-type") String content_type,
@Body FavMoviePost movie);
```

Se trata de la única llamada POST que se realiza a lo largo de la aplicación. Además de los parámetros que se introducen, idénticos a la de la anterior llamada mencionada, en esta llamada se incluirá una cabecera con el tipo de ítem que se está introduciendo (serie o película) y un objeto de cuerpo llamado FavMoviePost. Esta llamada también tiene de utilidad el hecho de que se puede marcar una película que ya era favorita como no favorita. La importancia de esto hace necesario el objeto mencionado antes, puesto que se incluirá en él una variable que indica si se desea marcar como favorito o como no favorito. Por último, se recoge un objeto SuccessResponse, que únicamente indicará si la llamada ha tenido éxito o no.

7.1.7.- Detalles de la cuenta

```
@GET("account")
Call<Account> getDetails(
@Query("api_key") String api_key,
@Query("session_id") String session_id);
```

Esta llamada GET tiene como utilidad única la recogida del nombre del usuario del sistema para darle un saludo cuando llegue al menú principal de la aplicación.

7.1.8.- Detalles de la película

```
@GET("movie/{id}")
Call<MovieDetails> getMovie(
@Path("id") int id ,
@Query("api_key") String apiKey);
```

Antes se mencionaba que, a la hora de recoger un objeto MovieResponse, se encontraba con un vector de películas con diversa información de cada una. Sin embargo, algunos detalles de las películas faltan, por ejemplo, los géneros a lo que pertenece. El uso de esta llamada GET es para paliar esos defectos que se recogen de otras llamadas y por eso se utiliza un objeto de salida distinto, MovieDetails, que contiene estos detalles mencionados.

7.1.9.- Autenticación de usuario invitado

```
@GET("authentication/guest_session/new")
Call<GuestSession> getGuestSessionID(
@Query("api_key") String api_key);
```

Esta llamada GET recoge un objeto GuestSession con la información necesaria para considerar al usuario como invitado, limitando algunas características de la aplicación

7.1.10.- Créditos de la película

```
@GET("movie/{movie_id}/credits")
Call<CreditsResponse> getCredits(
@Path("movie_id") int movie_id,
@Query("api_key") String api_key);
```

Como se indicaba en la llamada referida a los detalles de la película, esta llamada también se utiliza para recoger los créditos del ítem indicado y plasmarlos en la descripción individual del ítem. Para ello se encuentra de respuesta un objeto CreditsResponse que contiene toda la ficha artística y técnica de la película.

7.1.11.- Recomendaciones de TMDb

```
@GET("movie/{movie_id}/recommendations")
Call<MovieResponse> getTMDBRecommendations(
@Path("movie_id") int movie_id,
@Query("api_key") String api_key);
```

La última llamada que se realiza a la API de TMDb es utilizada para, en el caso en que la película seleccionada no se encuentre dentro de la base de datos de MovieLens, recoger una lista que la propia base de datos “TMDb” recomienda a partir del ítem seleccionado.

7.1.12.- Implementación de Retrofit

Todas las llamadas antes mencionadas carecerán de funcionalidad sin una correcta implementación del cliente REST. Para ello, se dispone de la clase en las que la aplicación muestra las películas más valoradas por TheMovieDatabase. Para ilustrarlo con un ejemplo, en primer lugar se muestran unas variables que se utilizarán a lo largo de la función que recoge los datos:

```
private static Retrofit retrofit = null;
private RecyclerView recyclerView = null;
private static List<Pelicula> movies = null;
private MoviesAdapter adapter = null;
public static final String BASE_URL =
"http://api.themoviedb.org/3/";
private final static String API_KEY =
"81c39ae2556fdc0f032d9a965252d8be";
```

La función connectAndGetApiData() es un ejemplo de cómo se implementa la llamada GET de Retrofit.

```
public void connectAndGetApiData(){
    if (retrofit == null) {
        retrofit = new Retrofit.Builder()
                .baseUrl(BASE_URL)
                .addConverterFactory(
                        GsonConverterFactory.create())
                .build();
    }
    MovieAPIService movieApiService = retrofit.create(MovieAPIService.class);
    Call<MovieResponse> call = movieApiService.getTopRatedMovies(API_KEY);

    call.enqueue(new Callback<MovieResponse>() {
        @Override
        public void onResponse(Call<MovieResponse> call,
Response<MovieResponse> response) {
            recyclerView = (RecyclerView) findViewById(R.id.recycler_view);
            recyclerView.setLayoutManager(new
GridLayoutManager(VistaPeliculaActivity.this, 3));
            movies = response.body().getResults();

            adapter = new MoviesAdapter(movies, R.layout.list_item_movie,
VistaPeliculaActivity.this, id_account, session_id);
            recyclerView.setAdapter(adapter);
            Log.d(TAG, "Number of movies received: " + movies.size());
        }

        @Override
        public void onFailure(Call<MovieResponse> call, Throwable throwable) {
            Toast.makeText(getApplicationContext(), "ERROR EN API",
                    Toast.LENGTH_SHORT).show();
            Log.e(TAG, throwable.toString());
        }
    });
}
```

Al ser llamada, se construye un objeto de tipo Retrofit nuevo denominada retrofit, que será estático y recogerá la información que recibe de la API para

posteriormente transformarla en el objeto deseado en la declaración de la llamada en la interfaz antes mencionada.

Se crea y se declara una llamada (“call”) que, de forma asíncrona, se encolará como un subproceso que esperará dos posibles escenarios: el escenario en el que se recibe una respuesta de la API y el escenario en que la comunicación falle.

En el caso de que se reciba una respuesta, RecyclerView conseguirá, mediante un adaptador del cual se profundizará más adelante, mostrar al usuario en el diseño de la vista la lista de películas que el objeto movies recoge de los resultados del cuerpo de la respuesta.

7.2.- Implementación como servicio web del sistema de recomendación

7.2.1.- Arquitectura del servicio web del sistema de recomendación en Azure

En esta sección, se ahondará más en el funcionamiento del sistema de recomendación en Azure y su implementación en la aplicación Android como API. A continuación, se presenta la arquitectura del servicio web y se procederá a detallar sus módulos.

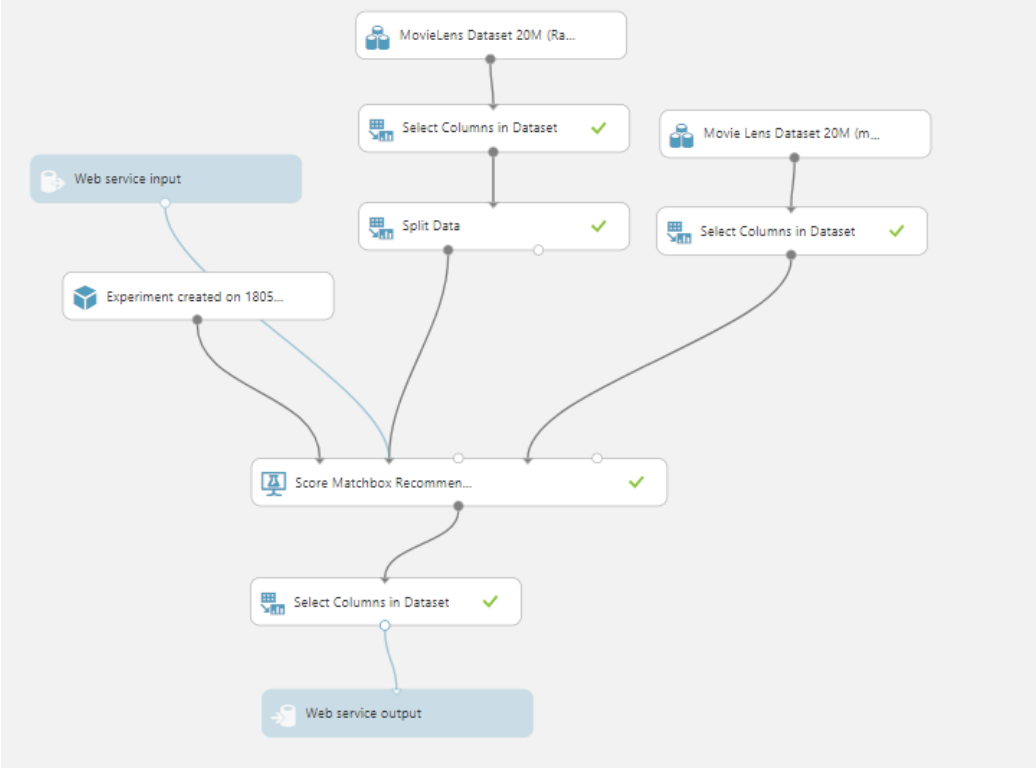

Figura 7.1: Arquitectura del servicio web del sistema de recomendación en Azure

Después de la creación del modelo de entrenamiento del sistema de recomendación, se procede a recibir lo que se denomina el "Experimento Predictivo".

En la figura 7.1, se puede observar como el módulo central de este experimento será el denominado "Score Matchbox Recommender". Si bien este tipo de módulo era utilizado en el capítulo anterior para ilustrar el funcionamiento y la arquitectura del "Experimento de entrenamiento", en esta arquitectura recibirá diferentes módulos de entrada.

Así, es posible observar que, como primer punto de entrada, recibirá el "Experimento de entrenamiento", detallado anteriormente. Posteriormente procederá a recibir las siguientes columnas de datos de los conjuntos de datos ratings.csv y movies.csv:

Figura 7.2: Selección de columnas de ratings.csv

Figura 7.3: Selección de columnas de movies.csv

Para a continuación, realizar la división de datos de las columnas seleccionadas de ratings.csv de manera idéntica a la que se realizaba en el anterior experimento.

A la hora de establecer el servicio web, se deberá introducir el módulo de servicio web, lo que causa que se active el módulo de producción, realizando las recomendaciones desde todos los ítems vistos durante el "Experimento de entrenamiento".

En cuanto a la configuración de este módulo central, se ha establecido la siguiente:

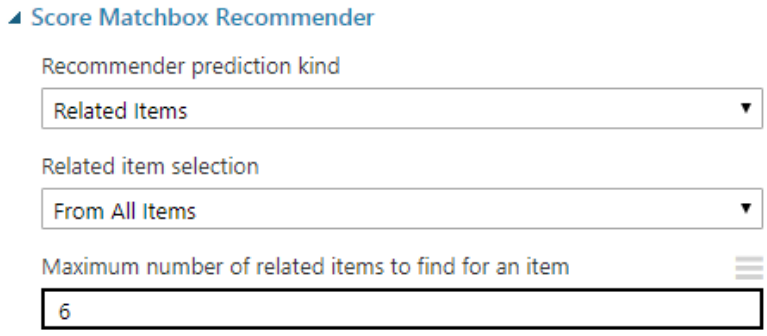

Figura 7.4: Configuración del módulo Score Matchbox Recommender

Esta configuración difiere de la utilizada para este tipo de módulo en el experimento anterior. La selección de ítems relacionados se ha establecido como "From All Items" y como se indica en la última figura, se contemplará una predicción de recomendaciones basadas en ítems relacionados. Esto contemplará una recomendación relacionando el ítem del que se quieren buscar recomendaciones (que se introduce con el Web Service input) con el resto de ítems que se encuentran en el modelo.

Así, a la salida de este módulo central, se podrá encontrar una lista de ítems relacionados, a los cuales se realizará una selección de columnas para recogerlos y guardarlos en un formato de salida apropiado para el servicio web.

7.2.2.- Establecimiento de servicio de cliente desde la aplicación

En cuanto al formato de los datos que se recibirán como entrada o se entregarán como salida del servicio web, será necesario enviar una petición POST al servicio API REST que ofrece Azure. De una forma similar a la que se han implementado las distintas llamadas relacionadas con el servicio web de TMDb, se ha implementado de la siguiente forma:

```
@POST("execute?api-version=2.0&details=true")
Call<ExecutionResults> getRecommendationsML(
@Header("Authorization") String api_key_ml,
@Header("Content-Length") int content_length,
@Header("Content-Type") String content_type,
@Body Inputs inputs);
```

Entre los parámetros utilizados, se ha utilizado la clave de API que otorga Azure como cabecera de autorización, la longitud del contenido del objeto Input que se pasará como cuerpo (y que contendrá el ítem del que se buscan recomendaciones) y el tipo de contenido, que en este caso, será un objeto JSON.

Por lo tanto, un ejemplo del objeto que se envía en la petición sería el siguiente:

```
{
 "Inputs": {
            "input1": {
              "ColumnNames": [
                "userId",
                "movieId",
                "rating"
              ],
              "Values": [
                [
                  "3212",
                  "2571",
                  "5"
                ],
                [
                  "3212",
                  "2571",
                  "5"                    ]
              ]
            }
 },
 "GlobalParameters": {}
}
```

Sin embargo, en su inicio se encontró un problema a la hora de abordar el envío del identificador de la película a Azure: los identificadores que se extraen de TMDb no coinciden con los identificadores de las películas que se encuentran en el sistema de recomendación en Azure. La única opción factible era, por tanto, utilizar la tabla que se encuentra en el fichero links.csv, el cual correlaciona los identificadores de las películas de MovieLens y los identificadores que se utilizan en el sistema de recomendación.

Para ello, se ha almacenado dentro de la aplicación un fichero SQLite, denominado linkslite.sqlite, el cual no es más que links.csv con un formato de este tipo. Así, antes de realizar la llamada POST al servicio web de Azure, se ha creado una clase llamada DBMovieLens.java, la cual contiene algunas funciones que permiten seleccionar y manipular, mediante comandos en lenguaje SQL, los identificadores buscados. Si se diera el caso en que no se encontrara el identificador en esa base de datos, la aplicación realizará la llamada al sistema de recomendación que posee TMDb mediante la llamada al servicio web que se ha mencionado en la sección anterior.

El ejemplo que se muestra a continuación, correspondiente a la función getRecommenderID que se puede encontrar en DBMovieLens.java, refleja la búsqueda del identificador. En este caso, cabe únicamente comentar que el uso del parámetro "semaphore" tiene como objetivo realizar esta búsqueda de correlación entre identificadores antes de enviar la película al sistema de recomendación (cuando el parámetro es igual a 0) en el que se busca el identificador de MovieLens, y después de recibir los identificadores de las recomendaciones (cuando el parámetro es igual a 1), donde lo que se buscan son los identificadores de TMDb que permitirán al usuario ver la información que alberga la base de datos de TMDb de esas películas.

```
public int getRecommenderID(int TMDbid, int semaphore){
    SQLiteDatabase db = this.getWritableDatabase();
    int result;
    Cursor cursor;
    try {
        if(semaphore==0) {
            cursor = db.rawQuery("select id from links where tmdbid = "
                    + TMDbid, null);
        } else {
            cursor = db.rawQuery("select tmdbid from links where id = "
                    + TMDbid, null);
        }
        if (cursor != null) {
            cursor.moveToFirst();
            if(cursor.getString(0)!=null) {
                result =
Integer.parseInt(String.valueOf(cursor.getString(0)));
                db.close();
                return result;
            } else {
                db.close();
                return -1;
            }
        }
    } catch (Exception e){
        Log.e("DB", e.toString());
    }
    db.close();
    return -1;
}

}
```

Habiendo recibido ya el identificador deseado, en el caso en que se encuentre en la base de datos de MovieLens, la aplicación se encargará de formar el objeto JSON deseado y se enviará mediante la llamada POST al servicio web de Azure.

Lo que se recibe a continuación es un objeto también JSON con un ejemplo de lo que se recibe enviando

```
{
 "Results": {
    "output1": {
        "type": "DataTable",
        "value": {
            "ColumnNames": [
                "Related Item 1",
                "Related Item 2",
                "Related Item 3",
                "Related Item 4",
                "Related Item 5",
                "Related Item 6"
            ],
            "ColumnTypes": [
                "String",
                "String",
                "String",
                "String",
                "String",
                "String"
            ],
            "Values": [
                [
                    "1445",
                    "473",
                    "4483",
                    "867",
                    "4686",
                    "4162"
                ],
                [
                    "1445",
                    "473",
                    "4483",
                    "867",
                    "4686",
                    "4162"
                ]
            ]
        }
        }
    }
}
```


Los valores que contendrá este objeto serán los identificadores de las películas que han sido recomendadas, los cuales volverán a introducirse en la función getRecommenderID con el objetivo de encontrar los identificadores de TMDb respectivos a cada uno.

Con esos identificadores ya será posible realizar llamadas al servicio web que ofrece TMDb y recoger la información, ficha técnica y artística de las películas y ofrecérselas al usuario.

Capítulo 8: Plan de pruebas

En este capítulo se presenta la fase del proyecto en la que se realizan las comprobaciones necesarias de la funcionalidad del sistema y de la aplicación.

Para este sistema serán necesarias diferentes comprobaciones para testar tanto el sistema de recomendación de Azure, probado en su propio entorno, y la aplicación Android desarrollada, en la que se utilizará el emulador que proporciona Android Studio.

8.1.- Entornos de pruebas

En primer lugar, se detallarán los dos entornos en los que se han probado los dos componentes de la aplicación.

Android Emulator es un emulador integrado en el entorno de desarrollo elegido para el proyecto, Android Studio, y se encarga de simular la aplicación Android en cualquier dispositivo seleccionable desde el emulador. Para ello, es necesario ser consecuente con el dispositivo que se escoja, puesto que cada uno tendrá una serie de características distintas, como las dimensiones de la pantalla.

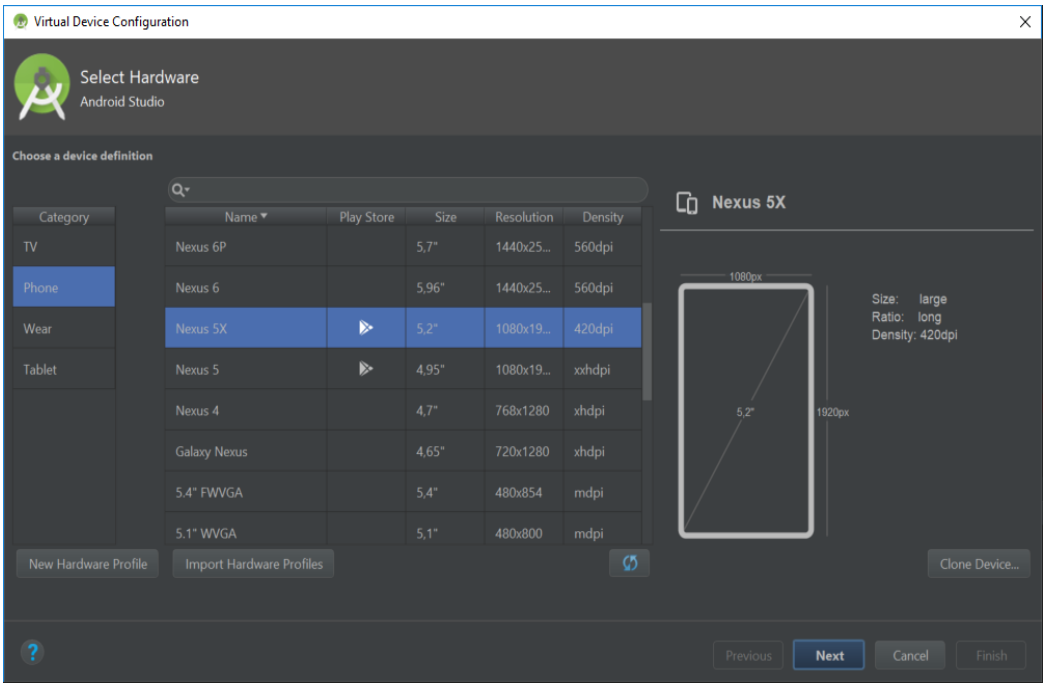

Figura 8.1: Captura de pantalla de la selección de dispositivo para Android Emulator

Para el plan de pruebas de este proyecto, después de un estudio sobre qué dispositivo puede ofrecer las características más óptimas para esta aplicación, el dispositivo elegido ha sido el Nexus 5X, con una visibilidad como la siguiente:

Durante esta dase, se ha realizado una selección adecuada de dispositivos dentro del conjunto con los que se desea que la aplicación sea compatible. Uno de ellos es el Nexus 5X:

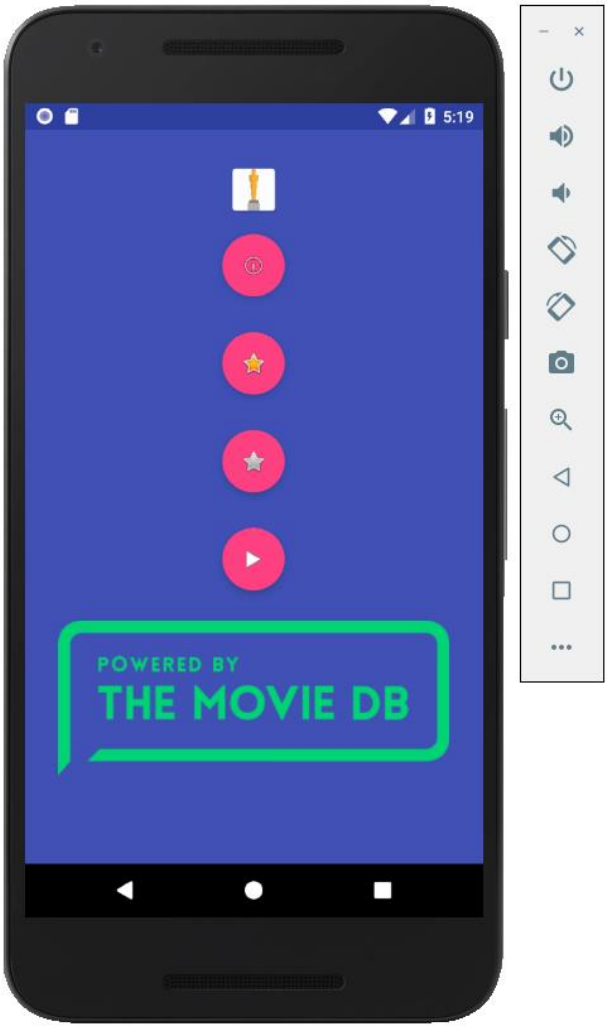

Figura 8.2: Android Emulator con Nexus 5X

En el caso del sistema de recomendación, el entorno de pruebas cambia inicialmente. Al montar el sistema de una manera independiente, se ha estudiado el entorno de Microsoft Azure Machine Learning y lo que puede ofrecer para testar un proyecto dentro del entorno de desarrollo. Por ello, se ha acabado concluyendo que el sistema más cómodo y óptimo a la hora de probar su funcionamiento era comprobando el servicio web de Azure mediante un formulario "Request/Response" que, en el entorno de Azure, abre un menú para ingresar los datos necesarios, como se presenta en la siguiente figura.

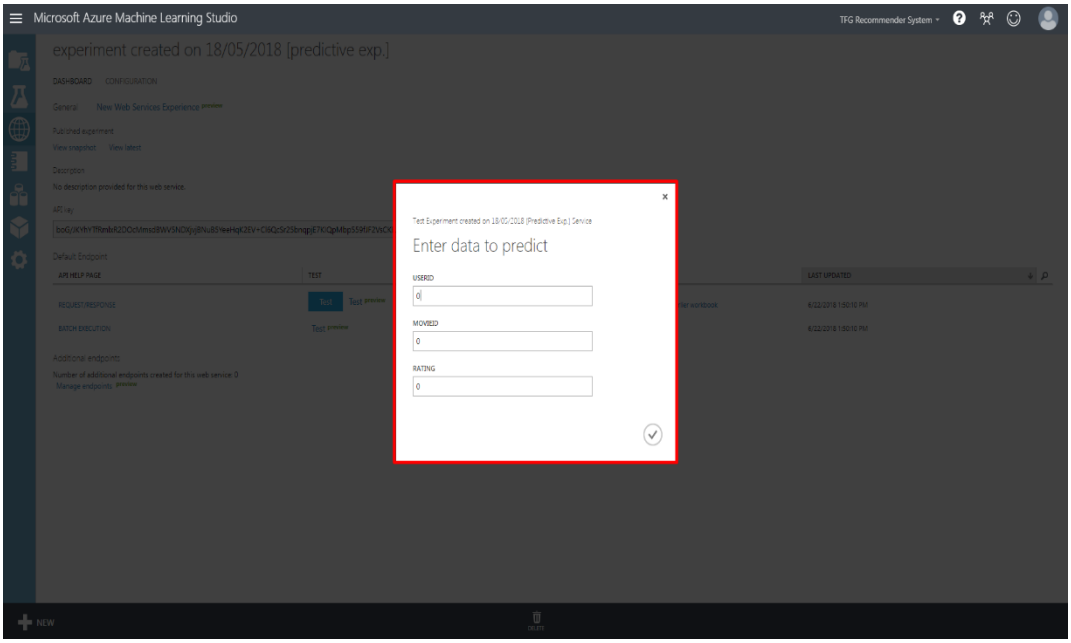

Figura 8.3: Captura de pantalla del entorno de prueba de Microsoft Azure

Para asimilarlo más a los datos que introduce el usuario cuando se encuentra en la aplicación, se procede a utilizar un identificador de usuario con valor 0, una valoración de 5 y el ítem a elegir. Finalmente en este entorno de prueba, se indica en al parte inferior del sistema la salida del servicio web:

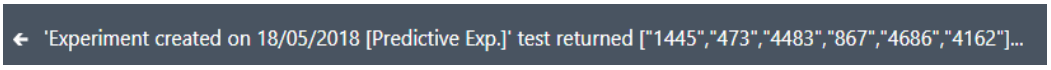

Figura 8.4: Recorte de la salida del servicio web desde Azure

8.2.- Formato de los casos de prueba

Para una óptima enumeración de los casos de prueba que se presentarán más adelante, se ha elegido un formato similar al utilizado para enumerar los requisitos de software del sistema. Para ello, se hará uso de tablas que contengan los siguientes campos:

- Número de identificación. Identificador único del caso de prueba con el siguiente formato: CP-XX.
- RS: Requisito funcional que comprueba el caso de prueba.
- Descripción de prueba: indicación de la funcionalidad del caso de prueba.
- Entrada de la prueba: acción, interacción o información que se introduce a la hora de la comprobación.
- Salida de la prueba: consecuencia de la prueba.

8.3.- Casos de prueba

Los casos de prueba se han realizado de forma secuencial y se disponen a continuación:

Número de identificación	CP-01
RS	RS-02 y RS-05
Descripción de prueba	Introducción de película de la que se desean las seis recomendaciones. Realizada en el entorno de prueba de Azure.
Entrada de la prueba	• Identificador de usuario (para nuestro caso, 0) • Identificador de la película • Valoración sobre 5 del ítem (para nuestro caso siempre será 5)
Salida de la prueba	• Los seis identificadores de las películas recomendadas a raíz del ítem introducido

Tabla 8.1: CP-01

Número de identificación	CP-02
RS	RS-06, RS-03 y RS-01
Descripción de prueba	El usuario entrará a la aplicación y recibirá un "request token". A continuación, aceptará dar permisos a terceras aplicaciones para el uso de sus datos en TMDb, introduciendo sus datos de sesión. A continuación recibe el identificador de sesión que le permitirá seguir navegando por la aplicación.
Entrada de la prueba	• Presionar botón de permisos. • Introducir nombre de usuario y contraseña • Aceptar los permisos • Presionar botón para recibir identificador de sesión • Continuar
Salida de la prueba	• Request Token • Session Id • Permisos para continuar como usuario.

Tabla 8.2: CP-02

Número de identificación	CP-03
RS	RS-11, RS-07 y RS-01
Descripción de prueba	El usuario, ya en el menú principal elegirá la opción de "Mis Películas Favoritas".
Entrada de la prueba	• Presionar botón "Mis películas favoritas".
Salida de la prueba	• Galería de películas disponibles • Título y poster de cada ítem dispuesto en la vista • Registro de películas favoritas en base de datos SQLite.

Tabla 8.3:CP-03

Número de identificación	CP-04
RS	RS-08. RS-01 y RS-11
Descripción de prueba	El usuario presiona una de los ítems que se presentan en la galería. Aparece una ficha técnica y artística de la película seleccionada.
Entrada de la prueba	• Elección de ítem
Salida de la prueba	• Ficha técnica de la película • Botón marcado como favorito (almacenamiento en tabla de películas favoritas) • Vista individual del ítem

Tabla 8.4: CP-04

Número de identificación	CP-05
RS	RS-01, RS-02, RS-05, RS-07, RS-09 y RS-12
Descripción de prueba	El usuario presiona el botón "Recomendaciones de Iván", que le lleva a una galería de películas recomendadas según TMDb o el sistema de recomendación desarrollado.
Entrada de la prueba	• Pulsar botón de recomendaciones.
Salida de la prueba	• Galería de Películas • Películas recomendadas por el sistema de recomendación de TMDb o Azure.

Tabla 8.5: CP-05

Número de identificación	CP-06
RS	RS-01, RS-07 y RS-10
Descripción de prueba	El usuario, ya en el menú principal, presionará el botón de buscar película. Ya en esa pantalla, introducirá el nombre del ítem a buscar y presionará el botón de buscar.
Entrada de la prueba	• Pulsar el botón de “Buscar Películas”. • Introducir título de la película a buscar • Pulsar botón de búsqueda
Salida de la prueba	• Galería de películas • Vista de películas junto con título y póster.

Tabla 8.6: CP-06

Capítulo 9: Conclusiones

En este capítulo, una vez terminada la implementación y el plan de pruebas del sistema, se exponen una serie de conclusiones extraídas a lo largo del desarrollo del proyecto. Así, a su vez, se marcan diversas indicaciones que, establecerán futuras líneas de trabajo.

9.1.- Conclusiones

De los requisitos planteados al inicio del proyecto, se ha conseguido una resolución completa y satisfactoria de los objetivos. El aprendizaje en máquina es una tecnología que lleva desarrollándose décadas, por lo que la formación recibida y la implementación inicial en Microsoft Azure, han resultado muy satisfactorias, punto clave que motivó el comienzo de este proyecto: el tratamiento de datos, las analíticas, crear un modelo, refinarlo y hacerlo accesible al usuario medio.

Otro de los hándicaps que se obtienen del proyecto es la formación en aplicaciones Android y servicios web REST. El mercado tecnológico de las aplicaciones está en auge, y cada día es más necesario en un entorno profesional conocer desarrollo de aplicaciones móviles. Por otra parte, el uso de tecnologías

de servicios REST es un punto clave en cualquier proyecto de desarrollo de software que se precie.

En resumen, se enumeran a continuación algunas de las conclusiones obtenidas a lo largo de la elaboración de este sistema:

- El *aprendizaje en máquina* es una tecnología clave a día de hoy y está marcando muchos proyectos tecnológicos. A consecuencia de estos sistemas, es posible la creación de aplicaciones útiles para negocios. Un ejemplo es la compañía francesa de marketing Criteo, la cual posee un motor de *machine learning* cuya función principal es la recomendación de productos de clientes a raíz de los datos de un usuario que navega por la red. Algo muy parecido a lo que se ha producido, a menor escala, en este proyecto.
- El uso de los servicios web de TMDb agiliza el desarrollo del proyecto, permitiendo esta asociación una serie de características muy interesantes, como la administración de roles, la autenticación o la administración de ítems entre ambos portales (el de la aplicación y el de la página web).
- Como dificultad añadida se ha de añadir la complicación que supone compaginar peticiones REST con las llamadas asíncronas que permite Android. Cabe recordar que cada actividad de la aplicación dispone de un hilo (UI Thread), encargada del funcionamiento y las respuestas a las peticiones del usuario. Así, la creación de diversos procesos que permitan que estas peticiones asíncronas se realicen en el segundo plano del hilo ha supuesto un hito en el desarrollo del proyecto.
- Como suplemento del punto anterior, la formación en desarrollo de aplicaciones móviles ha supuesto otro de los objetivos iniciales del proyecto. Puede resultar motivante la implementación de aplicaciones en este tipo de plataformas para iniciar formación en otras plataformas como iOS.

9.2.- Trabajos futuros

Acabado ya el desarrollo del proyecto, se presentan a continuación diferentes líneas de trabajo donde se puede continuar la senda que se ha abierto en este proyecto, incluyendo diversas mejoras que potenciarían el sistema de una forma notoria:

- **Mejora del diseño gráfico**: En las vistas que la aplicación ofrece al usuario se han estudiado diversas formas para que estas resulten más atractivas. Pese a lograr avances respecto a cómo mostrar galerías de películas o vistas individuales de ítems individuales, con la ayuda de un profesional en el diseño se lograría una mejora

importante en este tema. Así, el menú principal le resultaría más dinámico al usuario y permitiría una navegación más personal entre las características que ofrece la aplicación. Un diseñador gráfico aportaría su propia visión y estilo al diseño, acogiéndose a una mayor coherencia visual con la plataforma Android.

- **Sistema de recomendación nativa**: Pese a que la integración del sistema de recomendación en la nube era una idea inicial atractiva, la elaboración de un sistema propio sería otra de las mejoras a presentar. Sin embargo, la comunicación con este sistema cambiaría de forma drástica, por lo que se deberían estudiar diferentes algoritmos enfocados también en la colaboración y el contenido, como se ha desarrollado el sistema que se presenta en este trabajo. Un servicio de un tercero es capaz de adaptarse fácilmente a este tipo de aplicaciones y la forma de pago resulta más barata, pero un producto a medida añadiría esa personalidad que se comentaba en el punto anterior.
- **Gestión de datos propios**: A la hora de alimentar al sistema de recomendación se ha hecho uso de los conjuntos de datos que ofrecía MovieLens. Esto imposibilita la capacidad de monetizar los ingresos que podría recaudar la aplicación, ya que entre los términos y condiciones de uso de estos conjuntos de datos, se prohíbe la comercialización de cualquier aplicación que los utilice, por ser su enfoque es estudiantil y de investigación. Así, la opción óptima sería recolectar los datos de manera individual y formar una base de datos propia, completa y accesible de la misma manera que el equipo de MovieLens hizo.
- **Sistema de navegación con cines cercanos**: Al comienzo del diseño de la aplicación, una de las características que se planteó implementar era un sistema de navegación en el que se presentaran los cines cercanos al usuario con la cartelera disponible en la fecha en la que se busque. De esta manera, con datos actualizados y una librería amplia, se podrían ofrecer fichas técnicas de las películas que interesasen al usuario. La API de TMDb proporciona este tipo de datos e, integrada junto a la API que ofrece Google Places, proporcionaría una funcionalidad extra muy atractiva para los usuarios.
- **Desarrollo de aplicaciones para más plataformas:** Uno de los objetivos a cumplir con esta posibilidad es permitir la llegada de esta aplicación a más sectores del mercado, lo que provocaría un aumento de los beneficios. Así, se buscarían implementaciones en iOS y Windows Phone.
- **Integración de contenido audiovisual:** La última idea nace de la necesidad de ofrecer al usuario un avance de las posibles recomendaciones que la aplicación le ofrece según el ítem deseado. Así, una integración de contenido audiovisual, mediante el uso de

la API de TMDb o incluso la API que proporciona YouTube, serían dos opciones viables para implementar esta idea.

Capítulo 10: Marco Regulador

En este capítulo se presentan algunas de las leyes que se han tenido en cuenta a la hora del desarrollo del proyecto:

- **Ley General de la Comunicación Audiovisual[29].** Esta ley se encarga de legislar lo relativos los contenidos audiovisuales y las comunicaciones, con el objetivo de regular y ordenar el sector, la protección de los ciudadanos respecto a las posiciones dominantes del sector y de la restricción de acceso a contenidos universales. Esta ley se basa en la Directiva 2007/65/CE de la UE y potencian la búsqueda de normas que marquen la configuración de un régimen básico común, mediante garantías de los derechos de los consumidores y el pluralismo. En este proyecto, una de las mejores prácticas que se podría proponer respecto a esta ley es la de la etiquetación correcta de la aplicación con el objetivo de definir el público de la misma. De esta manera se buscaría cumplir el artículo 7 de esta misma ley, en la que se provee un marco para el control y restricción de contenidos a menores de edad, lo que implica que, si no se ha cumplido la mayoría de edad como usuario, se restrinja el acceso a películas no recomendadas para edades pequeñas.

[29] Ley 7/2010, General de la Comunicación Audiovisual. (BOE Nº 79, 1 enero 2010).

- **Ley de la Propiedad Intelectual[30].** Esta ley es la encargada de legislar la propiedad intelectual y todo lo relativo a ella. Se refiere a propiedad intelectual como el conjunto de derechos que corresponden a los propietarios, autores y artistas respecto a sus obras y prestaciones. En su imposición para este proyecto, se ha establecido una serie de mecanismos de protección que aparecen en su Libro III, Título I respecto a la infracción en los derechos. Por lo tanto, para el desarrollo de este proyecto y, en especial, su software, se deberá cumplir con lo establecido sobre el respeto a la propiedad intelectual del autor y de otras librerías de software utilizadas durante la implementación.
- **Ley Orgánica de Datos de Carácter Personal[31].** En cuanto al desarrollo de una aplicación móvil, como la que se presenta en este proyecto, se deben cumplir ciertos requisitos en cuando la protección respecto a lo que concierne el tratamiento de los datos personales y los derechos fundamentales de las personas físicas, sobre todo de su honor e intimidad. Así, todos los usuarios registrados en la página de "TMDb" y que hacen uso de esta aplicación habrán debido ser informados sobre sus derechos a permitir o ejercer el uso de estos datos desde aplicaciones de terceros, mostrando el objetivo de este uso.

[30] Real decreto legislativo 1/1996, Ley de Propiedad Intelectual. (BOE Nº 97, 22 abril 1996).
[31] Ley Orgánica 15/1999, de Protección de Datos de Carácter Personal. (BOE 298 , 14 diciembre 1999).

Capítulo 11: Entorno socioeconómico

Como ya se ha comentado en el capítulo 9, la capacidad de recolectar ingresos en la aplicación no es viable debido a la incapacidad para gestionar datos nativos. Una valoración propia de las películas y una recolección similar a lo ofrecido en MovieLens solucionaría este problema.

Profundizando en este tema, se debe estudiar el presupuesto que se recoge del capítulo 2:

Motivo del coste	Coste
Coste de Personal	9611,55 €
Coste de material	233,33 €
Software con amortización	88,88 €
Coste del servicio en la nube	60,76 €
Costes indirectos (20% de todo lo anterior)	1998,90 €
IVA (30% de todo lo anterior)	3598,03 €
TOTAL	15591,46 €

Figura 12.1: Presupuesto final del proyecto

MovieLens recoge la cantidad de 26 millones de valoraciones realizadas por 270.000 usuarios. La recogida de estos datos por nuestra parte incrementaría el presupuesto de nuestro proyecto, ya que sería recomendable la inclusión de un analista de datos encargado de recoger valoraciones optimizadas para nuestro sistema de recomendación. Esto llevaría a una incrementación en el coste de personal de en torno a 8000 euros.

Además de esta mejora, otra vía de monetización de esta aplicación sería dar acceso directo a los principales servicios de "video bajo demanda" para posibilitar al usuario la reproducción de las películas que le interesen. Sería interesante que, desde estas plataformas se fomentaran estas relaciones con aplicaciones relacionadas con el cine y que estén en auge para dar visibilidad a sus catálogos de películas.

Por último, a parte de estas mejoras es necesario mencionar la integración del servicio en la nube de Azure en una aplicación móvil como ejemplo de impacto en el contexto socioeconómico. Actualmente, multitud de compañías y desarrolladores apuestan por sistemas de recomendación nativos en sus aplicaciones móviles, lo que eleva el coste considerablemente. El servicio de Azure otorga al desarrollador la capacidad de manipular conjuntos de datos, estudiarlos para tu propio beneficio y explorar diferentes enfoques para solucionar un problema, sin que necesariamente el desarrollador tenga un alto conocimiento en el aprendizaje en máquina. Otro de los motivos que indica que es una gran elección es el bajo coste que tiene desplegar el propio sistema de recomendación en ese servicio. La funcionalidad de "*pay as you go*" que ofrece este tipo de servicios es sobresaliente y abarata costes respecto al despliegue de un sistema de recomendación nativo.

Lista de acrónimos

API: Application Programming Interface

ART: Android Runtime

BOE: Boletín Oficial del Estado

CDN: Content Delivery Network

CE: Constitución Española

CP: Caso de prueba

ERS: Especificación de requisitos de software

EU: Unión Europea

HTTP: Hypertext Transfer Protocol

IA : Inteligencia artificial

IEEE: Institute of Electrical and Electronics Engineers

IU: Instruction unit

MAE: Mean Absolute Error

ML: Machine Learning

NFC: Near Field Communication

POJO: Plain Old Java Object

REST: Representational State Transfer

RPC: Remote Procedure Call

SaaS: Software as a Service

SDK: Software Development Kit

SOAP: Simple Object Access Protocol

UTC: Universal Time Coordinated

XML: Extensible Markup Language

Bibliografía

[1] Gartner Says AI Technologies Will Be in Almost Every New Software Product by 2020. (2018). Retrieved from https://www.gartner.com/newsroom/id/3763265

[2] https://www.imdb.com/

[3] https://www.themoviedb.org/

[4] Mund, S. (2015). *Microsoft Azure machine learning*. Packt Publishing.

[5] "Dogs vs. Cats" en www.kaggle.com/c/dogs-vs-cats. Dogs vs. Cats | Kaggle. (2018). Recogido de https://www.kaggle.com/c/dogs-vs-cats

[6] Brink, H., Richards, J., & Fetherolf, M. (2017). *Real-world machine learning* (p. Chapter 1: What is Machine Learning). Shelter Island: Manning.

[7] Ilustración recogida de: Mund, S. (2015). *Microsoft Azure machine learning*. Packt Publishing.

[8] Jannach, D., Zanker, M., Felfernig, A., & Friedrich, G. (2010). *Recommender Systems: An Introduction*. Cambridge: Cambridge University Press. doi:10.1017/CBO9780511763113

[9] Lew, M., Sebe, N., Djeraba, C., & Jain, R. (2006). Content-based multimedia information retrieval. *ACM Transactions On Multimedia Computing, Communications, And Applications*, *2*(1), 1-19. doi: 10.1145/1126004.1126005

[10] Son, J., & Kim, S. (2017). Content-based filtering for recommendation systems using multiattribute networks. *Expert Systems With Applications*, *89*, 404-412. doi: 10.1016/j.eswa.2017.08.008

[11] Plummer, L. (2017). This is how Netflix's top-secret recommendation system works. Retrieved from http://www.wired.co.uk/article/how-do-netflixs-algorithms-work-machine-learning-helps-to-predict-what-viewers-will-like

[12] Gorakala, S. (2016). *Building Recommendation Engines*. Packt Publishing. ISBN: 978-1-78588-353-8

[13] Ilustración obtenida de: Gorakala, S. (2016). *Building Recommendation Engines*. Packt Publishing. ISBN: 978-1-78588-353-8

[14] http://www.apache.org/licenses/LICENSE-2.0.

[15] Imagen obtenida de: Cinar, O. (2015). *Android Quick APIs Reference* (p. 2). Berkeley, CA: Apress.

[16] Krajci, I., & Cummings, D. (2013). *Android on x86* (pp. 1-8). [New York, NY]: ApressOpen. doi: 10.1007/978-1-4302-6131-5

[17] Anderson J., University E., & Rainie L. (2005). Technical seminar report on cloud computing. *Intel Executive Summary*.

[18] Wang L., Ranjan R., Chen J. & Benatallah B. (2017). *Cloud Computing: Methodology, Systems and Applications*. CRC Press. ISBN: 978-1-351-83309-7

[19] Ilustración obtenida de: Wang L., Ranjan R., Chen J. & Benatallah B. (2017). *Cloud Computing: Methodology, Systems and Applications*. CRC Press. ISBN: 978-1-351-83309-7

[20] Copeland, M., Soh, J., Puca, A., Manning, M., & Gollob, D. (2015). *Microsoft Azure: Planning, Deploying, and Managing Your Data Center in the Cloud*. Berkeley, CA: Apress.

[21] Ilustración obtenida de: Copeland, M., Soh, J., Puca, A., Manning, M., & Gollob, D. (2015). *Microsoft Azure: Planning, Deploying, and Managing Your Data Center in the Cloud*. Berkeley, CA: Apress.

[22] Más información en developers.themoviedb.org

[23] Fielding, R. (2000). *Architectural styles and the design of network-based software architectures*. California: University of California, Irvine.

[24] IEEE STD 830-1998. "Especificaciones de los requisitos del Software", pág. 3.

[25] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872

[26] D. Stern, R. Herbrich y T. Graepel. *Matchbox: Large Scale Online Bayesian Recommendations.* Published in: *Proceedings of the 18th International World Wide Web Conference* 2009

[27] W. Chu and Z. Ghahramani. Gaussian processes for ordinal regression. Páginas 1019–1041, 2005.

[28] Figura extraida de: D. Stern, R. Herbrich y T. Graepel. *Matchbox: Large Scale Online Bayesian Recommendations.* Published in: *Proceedings of the 18th International World Wide Web Conference* 2009

[29] Ley 7/2010, General de la Comunicación Audiovisual. (BOE Nº 79, 1 enero 2010).

[30] Real decreto legislativo 1/1996, Ley de Propiedad Intelectual. (BOE Nº 97, 22 abril 1996).

[31] Ley Orgánica 15/1999, de Protección de Datos de Carácter Personal. (BOE 298 , 14 diciembre 1999).